Markt&Technik

BOOK-
WARE

AMIGA REFLECTIONS

Traumwelt und Realismus - Bilder im Raytracing-Verfahren auf dem Amiga

Auf Diskette enthalten:
Reflections, das schnelle, komfortable und leicht zu bedienende Raytracing-Programm.
Dazu im Buch eine ausführliche Bedienungsanleitung mit vielen Tips und Tricks.

BOOK-
WARE

AMIGA

Carsten Fuchs

REFLECTIONS

Traumwelt und Realismus – Bilder im Raytracing-Verfahren auf dem Amiga

Markt&Technik Verlag AG

CIP-Titelaufnahme der Deutschen Bibliothek

Fuchs, Carsten:
Amiga reflections : Traumwelt und Realismus – Bilder im Raytracing-Verfahren auf dem Amiga / [Carsten Fuchs]. – Haar bei München : Markt-u.-Technik-Verl., 1989
(Book-Ware)
ISBN 3-89090-727-X
NE: HST

Die Informationen in diesem Produkt werden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht.
Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.
Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen.
Trotzdem können Fehler nicht vollständig ausgeschlossen werden.
Verlag, Herausgeber und Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.
Für Verbesserungsvorschläge und Hinweise auf Fehler sind Verlag und Herausgeber dankbar.

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien.
Die gewerbliche Nutzung der in diesem Produkt gezeigten Modelle und Arbeiten ist nicht zulässig.

Commodore AMIGA ist eine Produktbezeichnung der Commodore Büromaschinen GmbH, Frankfurt, der ebenso wie der Name Commodore Schutzrechte genießt.
Der Gebrauch bzw. die Verwendung bedarf der Erlaubnis der Schutzrechtsinhaberin.
AMIGA ist ein eingetragenes Warenzeichen der Commodore-AMIGA Inc., USA
Sculpt 3-D ist ein eingetragenes Warenzeichen der Byte by Byte Corporation, USA
Silver ist ein eingetragenes Warenzeichen der Impulse Inc., USA

15 14 13 12 11 10 9 8 7 6 5
92 91 90

ISBN 3-89090-727-X

© 1989 by Markt&Technik Verlag Aktiengesellschaft,
Hans-Pinsel-Straße 2, D-8013 Haar bei München/West-Germany
Alle Rechte vorbehalten
Einbandgestaltung: Grafikdesign Heinz Rauner
Bildschirmfoto: Werner Leidl
Druck: Schoder, Gersthofen
Printed in Germany

Inhaltsverzeichnis

	Vorwort		7
Kapitel 1	**Überblick**		9
Kapitel 2	**Einführung**		13
Kapitel 3	**Construct**		15
	3.1	Überblick	15
	3.2	Erzeugung einfacher Körper	21
	3.3	Darstellungsarten	29
	3.4	Körper – Bausteine einer Szene	36
	3.5	Geometrische Veränderungen an Körpern	44
	3.6	Materialeigenschaften	49
	3.7	Texturen	54
	3.8	Beispiele zu Texturen	64
	3.9	Lichtquellen	71
	3.10	Laden und Speichern	74
	3.11	Beispiele	77
Kapitel 4	**Beams**		83
	4.1	Einleitung	83
	4.2	Vorverarbeitung durch Grid	84
	4.3	Raytrace-Parameter	85
	4.4	Speicherverbrauch und Rechenzeit	92

Kapitel 5 **Show** 95

5.1 Einleitung 95
5.2 Darstellungsarten beim Amiga 95
5.3 Show-Parameter 97
5.4 Get_IFF 99

Kapitel 6 **Objekt-Konverter** 101

Kapitel 7 **Raytrace-Verfahren** 103

7.1 Der Grundalgorithmus 104
7.2 Schatten 106
7.3 Spiegelung 108
7.4 Brechung 110
7.5 Suche_Objekt 113
7.6 Die Gittermethode 120
7.7 Der Gitterbaum 123
7.8 Die Beleuchtungsformel 126
7.9 Antialias 133
7.10 Begriffsdefinitionen 135

Kapitel 8 **Andere Verfahren zur Bilderzeugung** 141

8.1 Der Maler-Algorithmus 142
8.2 Der Tiefenpuffer-Algorithmus 143
8.3 Der Scan-Line-Kohärenz-Algorithmus 144

Kapitel 9 **Ausblick auf zukünftige Erweiterungen zu Reflections** 149

Literaturverzeichnis 151

Stichwortverzeichnis 153

Vorwort

Mit Reflections haben Sie ein Computergrafik-Programm erworben, das sehr realitätsnahe Bilder erzeugen kann. Die fotorealistische Bilderzeugung ist ein Teilbereich der Computergrafik, die in letzter Zeit durch Film und Fernsehen sehr populär geworden ist.

Computergrafik ist heute für jeden Computer fast schon selbstverständlich, zumindest im PC- bzw. Hobbybereich. Es läßt sich kaum noch ein Programm verkaufen, das nicht zumindest eine grafisch geführte Oberfläche besitzt. Dies war früher nicht so, da war in der Regel jede Ein- und Ausgabe am Computer textorientiert. Das hatte mehrere Gründe. Zum einen braucht man für Computergrafik sehr viel Speicher, und zwar um so mehr, je detail- und farbenreicher die Grafik sein soll. Und Speicher war früher sehr teuer. Ein weiterer Grund ist der relativ hohe Rechenaufwand. Schon eine fensterorientierte Benutzeroberfläche benötigt eine leistungsfähige CPU und hätte die früheren 8-Bit-Rechner doch überfordert. Die Spieleprogrammierer, die auch da schon weitgehend Grafik einsetzten und so schon Vorreiter für die 16-Bit-Rechner waren, operierten aber derart an den Grenzen der Rechner, wie es für den Normalbenutzer nicht möglich war. So hat sich im PC-Bereich die Computergrafik erst mit der Verbreitung der 16-Bit-Rechner allgemein durchgesetzt.

Die andere Seite war der Großrechner- und Forschungsbereich. Dort war Computergrafik schon seit den 60er Jahren Gegenstand intensiver Forschungen. Speicher und die nötige Rechenleistung war vorhanden oder wurde mit spezialisierten Grafikrechnern geschaffen. Besonders für Militär und Raumfahrt wurden damals schon Flugsimulatoren hergestellt, die aus heutiger Sicht natürlich astronomisch teuer waren. In der Zeit wurden die Grundlagen der meisten Verfahren erarbeitet, die heute noch benutzt werden.

Mit der steigenden Rechenleistung konnte die Computergrafik in den kommerziellen Bereich vordringen. Beim CAD (Computer Aided Design) wurde der Vorteil der visuellen Kontrolle eines konstruierten Gegenstandes erkannt.

Mit Raytracing (Strahlverfolgung) wurde der vorläufige Höhepunkt in der Bildqualität erreicht, bei Verwendung von genügend Farben ist ein so erzeugtes Bild kaum noch von einer Fotografie zu unterscheiden. Allerdings wurde diese Qualität mit einem weiteren

»Quantensprung« im Rechenaufwand erkauft. Anfangs waren nur die Superrechner in den Forschungslaboratorien in der Lage, solche Bilder herzustellen, und das auch nur bei stundenlanger Rechenzeit.

In den letzten Jahren wurden aber mit gutem Erfolg Datenstrukturen erforscht, die den Aufwand so reduzieren, daß Großrechner solche Bilder in Minuten und Rechner wie der Amiga solche Bilder immerhin in Stunden berechnen können.

Mit einer solchen Datenstruktur arbeitet auch Reflections. Reflections ist die jüngste Version einer ganze Linie von Raytrace-Programmen. Der Urahn, den ich während meines Studiums auf einem Apple II programmierte, rechnete noch mehrere Tage an einem Bild.

Als ich dann auf einen 16-Bit-Rechner umstieg, war für mich die Wahl nicht schwer, denn der Amiga erfüllt unter allen Rechnern in der Preisklasse als einziger die Voraussetzungen, die zum effektiven Arbeiten mit Raytrace-Programmen notwendig sind.

- Eine Voraussetzung sind viele Farben. Die 4096 Farben des Amiga sind in der Klasse noch unerreicht.
- Trotz aller Verbesserungen rechnet ein Raytrace-Programm immer noch stundenlang. Deswegen ist die Fähigkeit zum Multitasking beim Amiga sehr wichtig, ermöglicht sie doch weitgehend normal mit dem Rechner weiter zu arbeiten, während das Bild im Hintergrund berechnet wird. Ohne Multitasking wäre der Rechner immer stundenlang blockiert.

Ich möchte abschließend noch Herrn Rolf Peters sowie Herrn Jörg Klora für die Unterstützung danken, ohne die Reflections nicht entstanden wäre.

Und nun, viel Spaß bei der Erzeugung von Szenen und der Berechnung von Bildern. Lassen Sie Ihrer Phantasie freien Lauf.

Ihr Carsten Fuchs

Überblick

Reflections ist ein Programmsystem zur Erzeugung möglichst realistischer Bilder. Es verwendet dabei das Raytrace-Verfahren, ein Verfahren, das sehr rechenintensiv ist, aber bisher die besten Resultate liefert, nämlich Bilder, die fast schon Foto-Qualität besitzen.

Reflections ist ein Raytracing-Programmsystem, System deshalb, weil es aus mehreren Programmen aufgebaut ist. Diese Aufteilung in mehrere Programme ergibt sich durch die Teilprobleme, die bei der Erzeugung realistischer Computerbilder auftreten:

- Man möchte ein realistisches zweidimensionales Bild einer dreidimensionalen Szene bekommen. Diese Szene muß zuerst erstellt werden. Bei dieser Aufgabe werden Sie von dem Programm »Construct« unterstützt. »Construct« ist für die Erstellung einer 3-D-Szene genau das, was ein Texteditor zur Texteingabe oder ein Malprogramm zur Erzeugung eines 2-D-Bildes ist. Mit »Construct« können Sie eine Szene erstellen und verändern.

- Aus dieser Szene soll ein Bild berechnet werden, und zwar möglichst realistisch. Diese Aufgabe übernimmt das Programm »Beams«. Es verwendet das Raytrace-Verfahren, das heißt, Effekte wie Schatten, Spiegelung und Brechung können berücksichtigt werden. Nach der Eingabe der Parameter haben Sie damit nichts mehr zu tun. »Beams« läuft im Hintergrund und Sie können derweil Ihren Amiga anderweitig verwenden (falls der Speicher ausreicht).

- Raytracing ist ein vergleichsweise langsames Verfahren für die Bilderzeugung. In Kapitel 7 wird noch genau darauf eingegangen, woran das liegt. Durch eine geeignete Datenstruktur kann die Rechenzeit aber erheblich reduziert werden. Diese Datenstruktur wird vom Programm »Grid« erzeugt. »Grid« wird automatisch vor dem Start von »Beams« aufgerufen.

- Das errechnete Bild soll am Bildschirm dargestellt werden. Das wird von »SHOW« übernommen. Das Problem dabei ist, daß der Amiga trotz seiner enormen Grafikfähigkeit eigentlich noch nicht genug Farben hat, um das Bild so darzustellen, wie es

»Beams« berechnet. Daher rechnet »SHOW« das Bild für die diversen Grafik-Modi des Amiga um und versucht, die Farben möglichst gut anzupassen.

- Reflections ist ein Programmsystem, das heißt, die einzelnen Programme sind nicht unabhängig voneinander, sondern müssen beispielsweise in einer bestimmten Reihenfolge aufgerufen und mit Parametern versorgt werden. Dies wird von »Manager« besorgt. »Manager« überwacht zudem »Beams« bei seinem Rechendurchgang und informiert Sie, wie weit »Beams« mit seiner Berechnung ist.

Dies sind die Hauptbestandteile von Reflections. Auf der Programmdiskette ist Reflections folgendermaßen strukturiert:

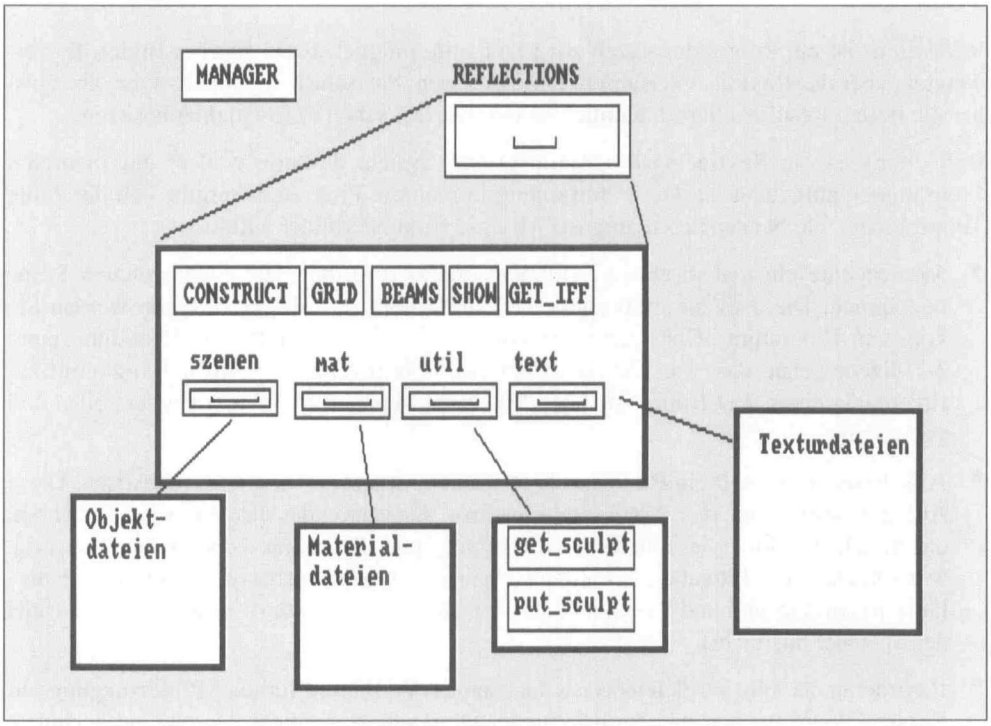


Bild 1.1: Die Struktur der Programmdiskette

Dort befinden sich zuerst einmal nur das Programm »Manager« und der Ordner »Reflections«; der Ordner wiederum enthält alle Programme und Daten. In Reflections liegen die Programme »Construct«, »Beams«, »Grid«, »SHOW«, »GET_IFF«. Zusätzlich enthält der »Reflections«-Ordner noch weitere Ordner:

szenen/ Dies ist der Standard-Ordner, in dem die Szenen liegen. Sie können Szenen aber auch an anderer Stelle speichern.

mat/ Hier werden Material-Dateien standardmäßig abgelegt.

util/ Hier liegen zwei Hilfsprogramme, um die Objekt-Dateien eines anderen Raytrace-Systems zu lesen bzw. zu schreiben.

text/ Hier liegen alle Texturdateien. Dies ist ein Ordner, der bei Reflections nicht nur Standard ist, sondern ein absolutes Muß. Alle Texturdateien müssen in diesem Ordner liegen.

Hier noch einige Bemerkungen zum Begleitbuch:

Das Buch ist sowohl zum Lesen in ruhiger Minute als auch zum Arbeiten am Rechner bzw. als Nachschlagewerk gedacht. Die Kapitel 7 und 8 enthalten Hintergrundinformationen zum Raytracing- und zu anderen Verfahren zur Bilderzeugung. Sie sind zum Arbeiten mit Reflections nicht unbedingt erforderlich. Nützlich ist es schon, wenn man genauer über die Arbeitsweise des Programms informiert ist. Kapitel 3 behandelt »Construct«, also den Teil, mit dem Sie als Benutzer am meisten zu tun haben. Dies sollten Sie also sorgfältig durchlesen. Hier sei noch besonders der Abschnitt über Textur erwähnt. Mit Textur kann der Benutzer mit sehr einfachen Mitteln interessante Effekte erzielen. Reflections enthält sehr vielfältige Arten von Texturabbildungen. Die erzielbaren Effekte belohnen für etwas Mühe in der Einarbeitung in die Materie.

Die Farbbilder 12 bis 15 deuten die Möglichkeiten von Reflections an. Besonders durch intensive Nutzung von Texturen gewinnen die Bilder einen starken Realitätsgrad. Hierbei sind dann aber auch Rechenzeiten zwischen fünf und zehn Stunden zu veranschlagen.

Einführung

Reflections als System ist recht komplex. Vor allem aber »Construct« mit seinen vielen Funktionen wird Ihnen einiges zu Lernen und Üben aufgeben. Deswegen soll zu Beginn eine kleine Einführung stattfinden, bei der zur Motivation einmal eine kleine Szene erzeugt und davon ein Bild gemacht wird.

Legen Sie die Reflections-Diskette in ein Laufwerk und klicken Sie am Bildschirm das Reflections-Icon an (Doppelklick). Daraufhin öffnet sich ein Kasten, in dem ein Reflections-Icon und ein »Manager«-Icon erscheinen. Klicken Sie den »Manager« an; über ihn haben Sie einen bequemen Zugang zu allen Programmen von Reflections. Am Bildschirm öffnet sich daraufhin ein leeres Fenster. Wenn Sie die rechte Maustaste drücken, können Sie am oberen Rand des Schirms die Menüpunkte des »Managers« sehen.

Wir wollen zuerst einmal eine kleine Szene erzeugen; dafür ist »Construct« zuständig. Wählen Sie also den Menüpunkt »Construct« an. Das Fenster füllt sich daraufhin mit einigen Gadgets, wovon im Augenblick nur das mit Namen »Start« interessiert. Klicken Sie es an; dadurch wird »Construct« gestartet. Nach einigen Momenten erscheint ein neuer Schirm, die Benutzeroberfläche von »Construct«.

Das Erzeugen einer Szene ist jetzt allerdings etwas schwierig, wenn Sie noch nicht in Kapitel 3 herumgeschnuppert haben. Allerdings gibt es eine kleine Test-Szene, die Sie laden können. Wenn Sie die rechte Maustaste drücken, erscheint das Menü von »Construct«. Wählen Sie im »Projekt«-Menü die Funktion »LoadSz«. Auf dem Schirm erscheinen diverse Schalter und Felder, mit denen Sie einen Dateinamen festlegen können. Sie müssen nur das Feld »Datei« anklicken und in der Box, die sich daraufhin öffnet, den Namen »test« eintippen (am Ende <Return>). Sodann klicken Sie »ok« an, und schon wird eine Test-Szene geladen.

Anschließend wählen Sie im »Projekt«-Menü die Funktion »LoadKamLi«. Jetzt brauchen Sie nur den »ok«-Schalter anzuklicken; der Name stimmt bereits.

Nun kann die Szene angeschaut werden. Wählen Sie im »Plotten«-Menü einmal die Funktionen »vorne«, »hinten«, »rechts«, »links«,... »oben«. Die Testszene wird jeweils aus der angegebenen Richtung dargestellt. Mit »Persp« erreichen Sie eine perspektivische Darstellung. So, wie Sie die Szene jetzt sehen, erscheint sie auch nach dem »raytracen«.

Gut, verändern wir nichts, sondern verlassen wir »Construct« mit »Progr.Ende« im »Projekt«-Menü. (»Construct« fragt, ob Sie sich sicher sind - Sie sind sich sicher!)

Wir finden uns im »Manager«-Fenster wieder. Um zum Menü des »Managers« zu kommen, klicken Sie dort den »Manager«-Schalter an. Im Menü wählen Sie jetzt »Beams«, das ist der Raytracer. Er erzeugt aus der Szene ein Bild. Das Fenster füllt sich jetzt mit Gadgets, um die Parameter für »Beams« festzulegen. Zuerst braucht »Beams« den Namen der Szene. Klicken Sie das »Szene«-Gadget an und tippen Sie dort »test« ein. Genauso verfahren Sie im »KamLi«-Gadget darunter. Hierunter wiederum geben wir an, wo das erzeugte Bild gespeichert werden soll. Geben Sie im Bild-Gadget »testbild« ein. Mit dem »Size«-Gadget legen wir die Bildgröße fest. Klicken Sie es einmal an, dann erscheint »test2«, eine Bildgröße von 64*52 Pixeln. Das genügt erst einmal als Testgröße. Anschließend klicken Sie einmal das »Schatt«-Gadget an, damit wird die Schattenberechnung eingeschaltet. Bedienen Sie dann das »Ray_Tie«-Gadget und tippen Sie eine »1« ein. Das wär's.

Mit dem »Start«-Gadget lösen Sie den Start aus. Zuerst erscheint eine seltsame Meldung: » → Grid «. Das bedeutet, daß zuerst »Grid« arbeitet, ein Programm, das zu der Szene eine Datenstruktur erzeugt, die »Beams« enorm hilft. Nach 1-2 Minuten ist »Grid« fertig, und die eigentliche Bilderzeugung startet. Sie können verfolgen, wie die 52 Zeilen nacheinander abgearbeitet werden. Nach kurzer Zeit ist »Beams« fertig und ruht wieder.

Jetzt wollen wir das Bild einmal anschauen. Klicken Sie »Manager« an, um wieder ins Menü zu kommen und wählen Sie dort »Show«. Wieder füllt sich das Fenster mit Gadgets. Tippen Sie im »Bild«-Gadget den Namen des erzeugten Bildes ein, nämlich »testbild«. Klicken Sie das »ViewMod«-Gadget einmal an, damit wird der farbenreiche HAM-Modus eingeschaltet; danach aktivieren Sie wieder »Start«.

Nach kurzer Zeit erscheint ein neuer Bildschirm, der sich in der linken oberen Ecke mit dem Bild füllt. Sobald Sie genug gesehen haben, klicken Sie einmal mit der Maus, und Sie sind wieder beim »Manager«. Der fragt, ob das Bild gespeichert werden soll. Wenn ja, sollten Sie zuvor im IFF-Gadget einen Namen eintippen, unter dem das Bild in einem besonderen Format (IFF) gespeichert wird. Ist das Bild erst einmal im IFF-Format gespeichert, so kann es mit »GET_IFF« (im »Manager«-Menü) sehr viel schneller geladen werden. Auch kann es so von anderen Programmen gelesen werden, da IFF ein Standard beim Amiga ist.

Die Schritte, die wir eben durchgeführt haben, werden auch in Zukunft der Weg sein, um ein Bild zu erzeugen. Zuerst in »Construct« eine Szene erzeugen, dann mit »Beams« ein kleines Testbild herstellen, mit »Show« ansehen. Meist ist man mit den ersten Versuchen nicht zufrieden und führt die Schritte mehrmals durch, bis endlich mit »Beams« ein großes Bild erzeugt wird. Dessen Berechnung wird schon relativ lange dauern.

Construct

3.1 Überblick

»Construct« ist ein grafischer Editor. So wie ein Texteditor zur Erstellung von Texten verwendet wird, kann mit »Construct« eine komplexe dreidimensionale Szene aufgebaut werden. Es stehen dabei Funktionen zum Erzeugen und Verändern von Szenen sowie zum Zugriff auf Teile der Szene zur Verfügung.

Was versteht man überhaupt unter einer Szene? Ganz allgemein gesagt, eine Menge von Objekten im dreidimensionalen Raum. Was aber sind Objekte? Da hat jeder seine eigenen Vorstellungen. Stellen Sie sich aber einmal beispielsweise ein Zimmer vor: In der Mitte ein Tisch, darum einige Stühle, vielleicht noch ein Sofa und in der Ecke ein Fernseher, an der Wand diverse Bilder und auf dem Tisch eine Weinflasche und ein paar Gläser.

All die aufgezählten Gegenstände könnte man durchaus als »Grundobjekte« betrachten und das dem Computer mitteilen. Aber abgesehen davon, daß diese Objekte jeweils schon recht kompliziert aufgebaut sind, müßten eine Unzahl von verschiedenen Alltagsgegenständen dem Programm bekannt sein, um die Phantasie der Benutzer auch nur halbwegs zu befriedigen. Diese Lösung ist nicht praktikabel, außerdem sind solch komplexe Objekte für den Raytracer nicht sehr effektiv. Der rechnet am liebsten mit sehr einfachen Objekten.

Reflections kennt daher als Objekte nur Dreiecke und Kugeln. Für Sie als Benutzer ist das aber nicht sehr angenehm. Sie haben eine Szene mit den obengenannten Gegenständen im Kopf und nicht eine Unmenge von Dreiecken.

Hier schließt »Construct« die Lücke. »Construct« hilft Ihnen, die Gegenstände Ihrer Vorstellung aus den Grundobjekten Dreieck und Kugel zusammenzusetzen und dem ganzen einen Namen zu geben.

Beispiel: Sie erzeugen mit den diversen Tools aus »Construct« einen Fernseher zum Beispiel aus 500 Dreiecken und geben dieser Dreiecksmenge den Namen »Fernseher«. Sie

müssen sich jetzt nicht die Dreiecke merken, wenn Sie sie weiterverarbeiten wollen. Sie teilen »Construct« nur den Namen dieser Dreiecksmenge mit und »Construct« vollführt die gewünschte Operation an allen Dreiecken, die zu »Fernseher« gehören.

An dieser Stelle ist es angebracht, auf Bezeichnungen einzugehen, die in »Construct« und Reflections gelten, in »Construct« sogar eine zentrale Bedeutung haben:

Objekt: Mit Objekt sind immer die Grundobjekte Dreieck oder Kugel gemeint.

Körper: Ein Körper ist eine beliebige Teilmenge aller Objekte, die »Construct« kennt, verbunden mit einem Namen, unter dem diese Teilmenge ansprechbar ist. Im obigen Beispiel ist »Fernseher« der Name eines Körpers, der aus 500 Dreiecken zusammengesetzt ist. Jede Operation oder Funktion in »Construct« wird an Körpern vorgenommen. Sie als Benutzer verschieben, drehen, vergrößern oder verzerren also keine Dreiecke, sondern Körper wie etwa »Fernseher«, »Stuhl«, »Sofa«, »Flasche« oder »Glas«.

So, damit haben Sie eine Szene geschaffen. Doch bisher war nur von der Geometrie der Szene die Rede: Wo sind welche Objekte im dreidimensionalen Raum? Das genügt aber noch nicht zum Raytracing.

Als weitere Komponente spielen die »Materialeigenschaften« eine Rolle. Körper sind ja zusätzlich zu ihrer Position auch noch durch Farbe und Oberflächenbeschaffenheit gekennzeichnet. Reflections hat eine Bibliothek von Materialien, die Sie an die Körper vergeben können. Sie können damit etwa dem »Sofa« das Material »Schwarzes Leder« zuteilen, oder dem »Tisch« das Material »Holz«. Zusätzlich können Sie sich aber auch noch eigene Materialien schaffen und in die Bibliothek einfügen. Die Materialbibliothek beinhaltet auch Texturen. Texturen sind einfach Muster, die als IFF-Bild vorliegen. Reflections besorgt automatisch die richtige Abbildung der Muster auf die Objekte.

Funktionen von »Construct«

»Construct« ist Menü-orientiert. In einem Hauptmenü sind die Funktionen aus »Construct« in sieben Gruppen aufgeteilt. Die Funktionen werden in den folgenden Kapiteln ausführlich erklärt, hier kommt schon mal ein Kurzüberblick, der auch zum Nachschlagen benutzt werden kann.

Das **Hauptmenü** hat sieben Menüpunkte:

1) *Projekt:* Lesen und Schreiben der benötigten Files. Programmbeendigung.
2) *Plotten:* Alle Funktionen zum Darstellen von Körpern, Verändern der Darstellungsparameter.
3) *Körper:* Alle Funktionen zur Verwaltung von Körpern.
4) *Geomet:* Geometrische Veränderungen an Körpern.
5) *Material:* Verwaltung und Manipulation von Materialeigenschaften.
6) *Tools:* Funktionen zur Erzeugung von Grundkörpern.
7) *Licht:* Verwaltung und Manipulation von Lichtquellen, Hintergrund und Umgebungslicht.

Nun einige Stichworte zu den Untermenüpunkten.

Menü Projekt:

- *LoadSz:* Laden von Objekt-Files.
- *SaveSz:* Speichern von Objekt-Files.
- *LoadKam/Li:* Laden von Kamera/Licht-Files.
- *SaveKam/Li:* Speichern von Kamera/Licht-Files.
- *LoadMat:* Laden von Material-Files.
- *SaveMat:* Speichern von Material-Files.
- *Progr.Ende:* Programmabbruch.

Menü Plotten

- *Körper:* Bestimmung eines Körpers, der geplottet werden soll.
- *Fenster:* Einstellung des Darstellungsbereiches, so daß der Plotkörper vollständig im Plotfenster erscheint.
- *Vorne:* Darstellung des Plotkörpers von vorne.
- *Hinten:* " hinten.
- *Rechts:* " rechts.
- *Links:* " links.
- *Unten:* " unten.
- *Oben:* " oben.
- *Persp:* Darstellung des Körpers aus Blickrichtung der Kamera.
- *Plotten:* Plotten des Körpers, so wie zuletzt eingestellt.
- *Bild*2:* Verdoppelung des Darstellungsbereiches.
- *Hidd_0:* Plotten aller Objekte des Körpers.
- *Hidd_1:* Plotten der Objekte, die dem Betrachter zugewandt sind.
- *Hidd_2:* Plotten aller Objekte. Kennzeichnung der abgewandten Objekte.
- *Farb_1:* Einfarbig Plotten.
- *Farb_2:* Objekte farbig nach Materialien kennzeichnen.
- *Kamera:* Ermittelung der Kamera-Grundposition.
- *Kam.beweg:* Bewegen der Kamera.

Menü Körper:

- *zeigen:* Informationen über Objektanzahl und Materialien des Körpers.
- *AddNeu:* Zusammenfassung von Körpern zu einem neuen.
- *Add:* Erweitern eines Körpers.
- *Sub:* Abziehen von Körpern.
- *doppeln:* Verdoppelung von Körpern.
- *define:* Grafische Definition neuer Körper.
- *Material:* Vergeben von Materialeigenschaften an Körper.
- *K_Löschen:* Löschen von Körpern.
- *O_Löschen:* Löschen von Körpern und ihren Objekten.

- *Schneiden:* Heraustrennen von Körpern aus anderen Körpern.
- *Rund:* Rundung der Objekte eines Körpers.
- *Eckig:*
- *Def_Mat:* Definition eines Körpers durch Materialeigenschaften.
- *Umdreh:* Umkehrung der Orientierung aller Dreiecke eines Körpers.

Menü Geomet

- *verändern:* Verschieben, vergrößern, verkleinern, verzerren von Körpern.
- *drehen:* Rotieren von Körpern.
- *knittern:* Fraktalisieren von Körpern.

Menü Material

- *NeuMat:* Einrichtung einer neuen Materialeigenschaft.
- *EditMat:* Veränderung einer Materialeigenschaft.
- *Matweg:* Löschung einer Materialeigenschaft.
- *NeuOb:* Einrichtung einer neuen Oberflächenbeschreibung.
- *EditOb:* Verändern einer Oberflächenbeschreibung.
- *Obweg:* Löschung eine Oberflächenbeschreibung.
- *NeuTex:* Einrichtung einer neuen Textur.
- *EditTex:* Verändern einer Textur.
- *Texweg:* Löschen einer Textur.

Menü Tools:

- *Rot.Körp:* Erzeugung eines Rotationskörpers.
- *3D-Polygon:* Erzeugung eines Polygons (flach oder 3D).
- *Zylinder:* Erzeugung eines Zylinders.
- *Kegel:* Erzeugung eines Kegels.
- *Kugel:* Erzeugung einer Kugel.
- *Quader:* Erzeugung eines Quaders.
- *Schnitt.Krp:* Erzeugung eines Schnittkörpers.

Menü Licht

- *NeuLicht:* Einrichtung einer neuen Lichtquelle.
- *Lichtfarbe:* Ändern der Lichtfarbe.
- *LichtPos:* Veränderung der Lichtposition.
- *Lichtweg:* Löschung einer Lichtquelle.
- *Hintergrund:* Ermittelung des Hintergrundmaterials.
- *Umgebung:* Eingabe des Umgebungslichtes.

Benutzerschnittstelle

»Construct« ist ein interaktives Programm. Es muß sehr häufig mit dem Benutzer kommunizieren. Es erhält von ihm Kommandos und Parameter, liefert ihm aber auch Informationen, vor allem grafische. Seine Kommandos erhält »Construct« über Menüs, aus denen der Benutzer die gewünschte Funktion mit der Maus wählt. Für viele Funktionen sind aber weitere Angaben nötig.

Einige Eingabemöglichkeiten kommen in vielen verschiedenen Funktionen vor und werden daher hier kurz angeschnitten:

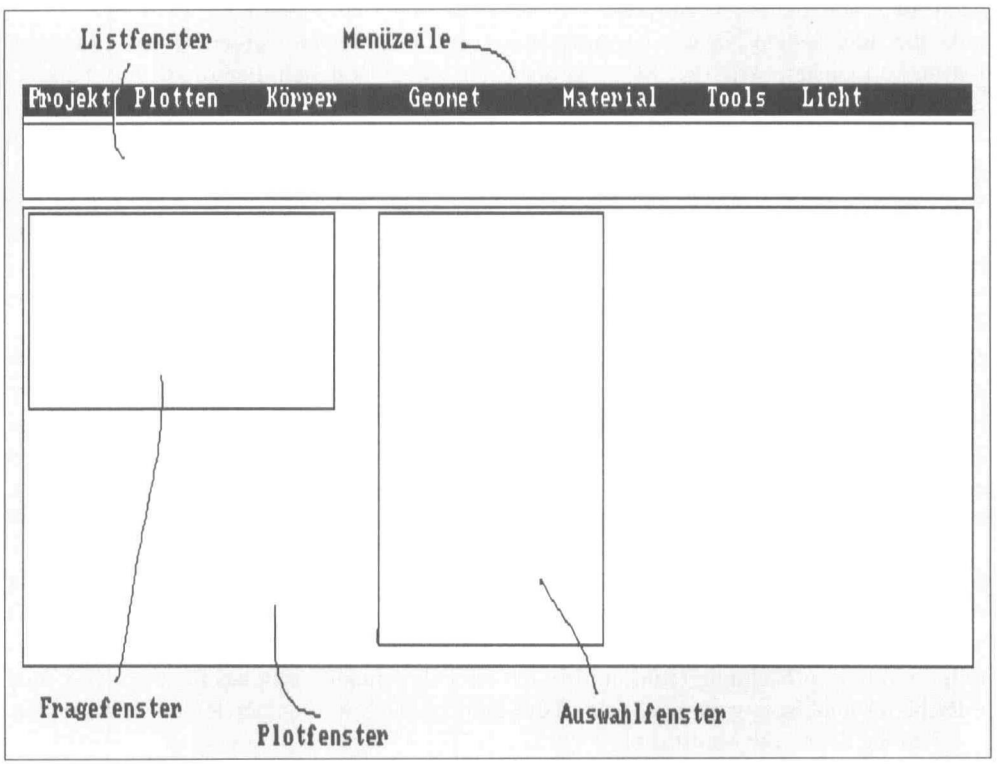


Bild 3.1: Die Fenster von »Construct«

1) Plotfenster

Im Plotfenster erfolgen alle grafischen Ausgaben von »Construct«, das heißt, alle Objekte werden im Plotfenster gezeichnet. Außerdem manipulieren Sie die Körper im Plotfenster mit Hilfe der Maus. Wenn das Plotfenster aktiv ist, erscheint als Mauszeiger anstelle des Pfeils ein Fadenkreuz.

2) Listfenster
Das Listfenster liegt oberhalb des Plotfensters und dient für Textausgaben von »Construct«. Dort erscheinen hauptsächlich Anweisungen von »Construct«, aber auch numerische Daten oder Auflistungen von Materialnamen. Das Menü von »Construct« ist mit dem Listfenster verbunden. Das Listfenster ist immer aktiv, wenn der normale Mauszeiger zu sehen ist. Falls sie einmal nicht auf das Menü zugreifen können, müssen Sie nur kurz ins Listfenster klicken (linke Maustaste). Dann ist das Menü wieder erreichbar.

3) Auswahlfenster
Das Auswahlfenster erscheint in der Mitte des Bildschirms, wenn »Construct« eine Auswahl zwischen mehreren Körpern, Dateien oder Materialien benötigt, zwischen Dingen also, die durch einen Namen gekennzeichnet sind. Die Namen werden in dem Auswahlfenster aufgelistet. Mit der Maus können Sie einen Rahmen innerhalb des Fensters bewegen und dadurch einen Namen kennzeichnen. Klicken Sie dann mit der linken Maustaste, so haben Sie den Namen gewählt, der gerade eingerahmt ist. Klicken Sie mit der rechten Taste oder befindet sich der Rahmen über keinem Namen, wird der Auswahlvorgang abgebrochen. Sind mehr Namen vorhanden, als ins Fenster passen, können Sie die Namen »hochrollen« lassen, indem Sie den Mauszeiger unter das Auswahlfenster bewegen und »niederscrollen«, wenn Sie den Mauszeiger über das Fenster positionieren.

4) Rechteck
Das Rechteck dient der Eingabe von zweidimensionalen Parametern, zum Beispiel der Verschiebung eines Körpers in x- und y-Richtung, aber auch vielen anderen Möglichkeiten, die hier nicht alle aufgezählt werden können. Benötigt »Construct« zweidimensionale Daten, dann erscheint auf dem Plotfenster ein Rechteck. Dieses Rechteck können Sie a) verschieben und b) in Größe und/oder Form verändern. Das Verschieben erfolgt, indem Sie den Mauszeiger ins Rechteck bewegen; sobald der Zeiger im Rechteck ist, folgt es den Mausbewegungen. Vergrößern können Sie durch Drücken der linken Maustaste. Solange die Taste 0 gedrückt ist, bleibt das Rechteck an seiner Position, verändert aber seine Größe entsprechend der Mausbewegung. Bei Loslassen der Taste folgt das Rechteck wieder, wie zuvor, der Maus. Manche Funktionen erfordern eine unabhängige Vergrößerung in x- und y- Richtung (Bildschirm). In dem Fall verändert sich das Rechteck in x- und y-Richtung unabhängig, je nach Mausbewegung. Sind Sie mit dem Rechteck zufrieden, drücken Sie die rechte Maustaste.

5) Proportionalregler
Bei Eingabe von numerischen Daten erscheint der Proportionalregler an der Unterkante des Plotfensters. Es ist ein waagerechter weißer Balken, innerhalb dessen eine senkrechte Marke der Mausbewegung folgt. Im Listfenster wird immer der numerische Wert angezeigt, der der Position der Marke entspricht. Haben Sie den richtigen Wert getroffen, drücken Sie die linke Maustaste.

6) Frage-Fenster
Dieses Fenster erscheint in der linken oberen Ecke des Plotfensters immer dann, wenn »Construct« für eine laufende Funktion eine Entscheidung benötigt. Dies können einfache Ja/Nein-Entscheidungen sein, oder auch die Eingabe eines Namens (für einen Körper oder eine Datei etc.). Häufig erscheint dann im Listfenster zusätzlich ein erläuternder Text.

7) Datei-Auswahlbox
Die Box erscheint immer, wenn Sie Dateien laden oder speichern wollen, wenn also Dateinamen festzulegen sind. Eine Beschreibung dieser Box folgt in Abschnitt 3.10.

Starten von »Construct«

Normalerweise wird »Construct« vom »Manager« aus gestartet. Im »Manager«-Menü wird die Funktion »Construct« gewählt. Daraufhin füllt sich das »Manager«-Fenster mit einigen Gadgets, um die Parameter von »Construct« festzulegen. Oben ist das »Farben«-Gadget. Durch Anklicken können Sie zwischen 4, 8 und 16 Farben wählen. Je mehr Farben Sie verwenden, desto mehr Speicher wird benötigt. (Die Farben werden meist erst bei sehr umfangreichen Szenen gebraucht.) Darunter sind zwei Gadgets, mit denen Sie »Construct« mitteilen, für wie viele Punkte und Objekte er Speicher reservieren soll. 1000 Punkte und Objekte, wie es standardmäßig eingestellt ist, reichen für den Anfang auf jeden Fall aus. Wenn Sie zu viele Punkte und Objekte benutzen wollen, reicht eventuell der Speicher nicht. Mit dem »Start«-Gadget lösen Sie den Start von »Construct« aus.

»Construct« kann aber auch vom CLI aus gestartet werden. Nach dem Aufruf

```
Ref:Reflections/Construct
```

fragt »Construct« nach Anzahl Farben, Punkten und Objekten. Ihre Angaben müssen Sie dann eintippen.

3.2 Erzeugung einfacher Körper

In diesem Abschnitt befassen wir uns mit den Funktionen, die »Construct« zur Erzeugung einfacher Körper zur Verfügung stellt. »Einfache Körper« ist vielleicht ein etwas irreführender Begriff, denn man kann hier schon recht komplexe Gebilde schaffen, nur soll dadurch angedeutet werden, daß diese Körper die Grundbausteine sind, auf denen die weitere Gestaltung der Szene aufbaut. Alle Funktionen sind im »Tools«-Menü zusammengefaßt und beinhalten folgende Punkte:

- *Rotat.Körper*
- *3D_Polygon*
- *Zylinder*
- *Quader*
- *Kegel*
- *Kugel*
- *Schn.Körper*

Einige der Funktionen erklären sich schon von ihrem Namen her. Zunächst werden wir die einfachen behandeln. Die Reihenfolge ist demnach anders als im Menü:

Zylinder

Mit »Zylinder« können Sie einen Zylinder erzeugen. Der Zylinder wird selbstverständlich aus Dreiecken zusammengesetzt. Bei Anwahl von Zylinder erscheint im Plotfenster ein Rechteck. Dieses soll den Zylinder von der Seite darstellen bzw. seine Silhouette. Mit dem Mauszeiger können Sie nun das Rechteck bewegen und in seiner Form verändern. Bei einem dünnen und hohen Rechteck wird der Zylinder zu einer flachen Scheibe, bei einem breiten und flachen ein dünner Stab. Durch Probieren bekommen Sie schnell ein Gefühl für die Form des Zylinders. Bild 3.2 zeigt, wie verschiedene Rechteckformen das Aussehen des Zylinders beeinflussen.

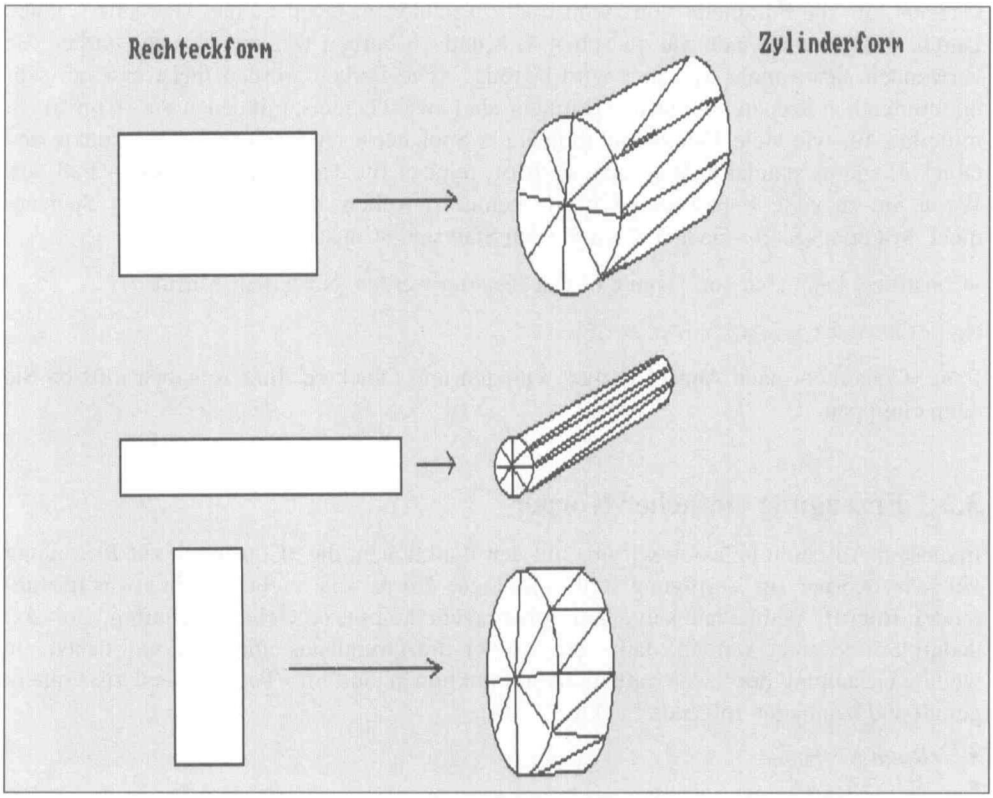


Bild 3.2: Rechteckform und Zylinderform

Sind Sie mit dem Rechteck fertig, möchte »Construct« die Anzahl der »Winkelschritte« wissen. Geben Sie die über den Proportionalregler ein. Die Anzahl der Winkelschritte gibt die Qualität der Rundungen des Zylinders an. Je größer die Anzahl, desto eher erhält man einen runden Zylinder, aber desto mehr Dreiecke werden gebraucht. Normalerweise hat man bei einer Anzahl zwischen 10 und 20 Winkelschritten schon einen annähernd runden Zylinder. Danach müssen Sie nur noch einen Namen für den Zylinder eingeben, und schon können Sie mit ihm arbeiten. Die Lage des Zylinders ist an der Ost-West-Achse orientiert.

Quader

Hiermit wird ein Würfel oder ein Quader erzeugt. Sobald Sie »Quader« anwählen, erscheint wie bei »Zylinder« ein Rechteck, mit dem Sie die Form des Quaders bestimmen. Ändern Sie das Rechteck nicht, wird es ein Würfel, oder Sie können ein flaches Brett oder einen langen Stab erzeugen. Wenn Sie mit dem Rechteck fertig sind, muß nur noch dessen Name eingetippt werden. Sie können mit dem Rechteck nur die Höhe und die Breite des Quaders bestimmen, die Tiefe bleibt unverändert.

Kegel

Bei dieser Funktion erscheint auf dem Plotfenster ein gleichschenkliges Dreieck, dessen Form durch Bewegung des Mauszeigers verändert werden kann. Das Dreieck soll die Silhouette des Kegels darstellen. Sind Sie mit der Form zufrieden, drücken Sie die linke Maustaste. Wie bei »Zylinder« will »Construct« jetzt auch die Anzahl der Winkelschritte wissen. Danach noch den Namen eingetippt – fertig ist der Kegel!

Auch der Kegel ist mit der Spitze nach Westen und dem Boden nach Osten orientiert. Bild 3.3 zeigt eine Zuordnung von Dreiecksformen zu Aussehen des Kegels.

Kugel

Bei »Kugel« brauchen Sie, außer dem Namen, nichts einzutippen, denn die Form steht fest und Größe und Position kann an anderer Stelle verändert werden. Nur den Namen der Kugel möchte »Construct« wissen.

3D-Polygon

Die Hauptaufgabe ist hier die Erzeugung eines Polygons, eines Vielecks. Da »Construct« aber nur mit Dreiecken rechnet, wandelt es das von Ihnen definierte Polygon in eine Vielzahl von Dreiecken um. Es »trianguliert« das Polygon. Zusätzlich kann es in die Tiefe gezogen werden. Damit erhalten Sie einen kompakten Körper anstatt einer Fläche. Außerdem können in das Polygon Löcher eingefügt werden; das sind wieder Polygone, die später beim Triangulieren ausgespart werden. Die Löcher werden insbesondere dann gebraucht, wenn man dreidimensionale Buchstaben erzeugen will. Für ein »O« oder ein »A« braucht man die Löcher unbedingt (siehe unten).

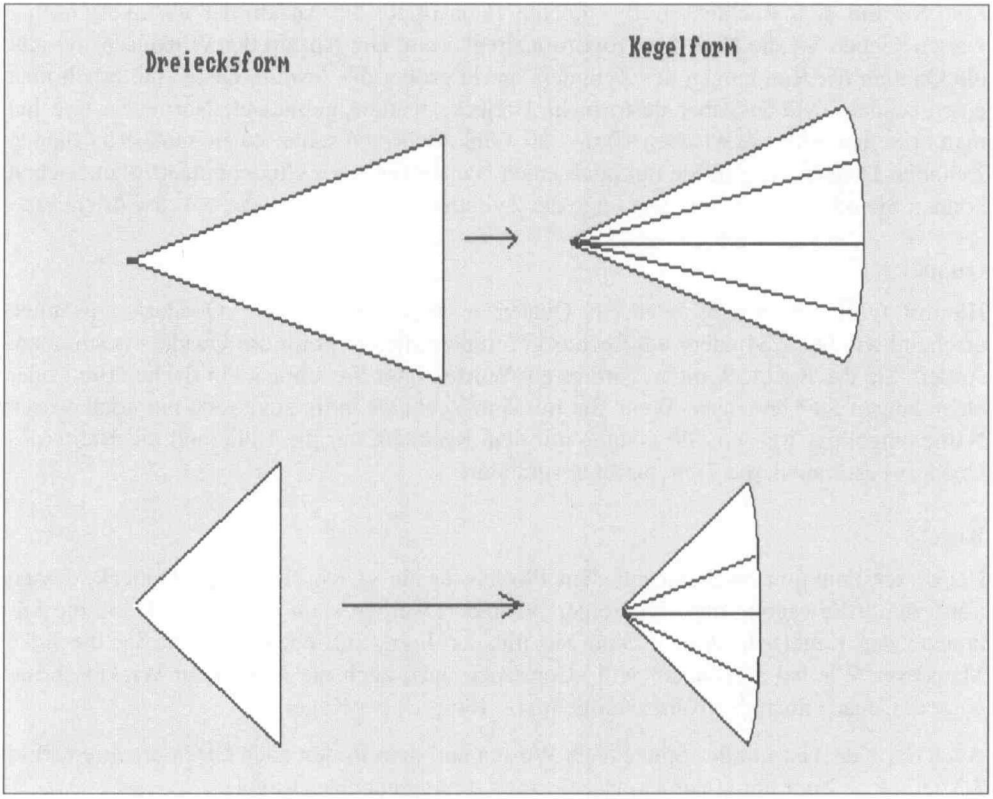


Bild 3.3: Dreiecksform und Kegelform

Bei Anwahl von Polygon erscheint auf dem Plotfenster ein Gitter. Mit der Maus können Sie dann ein Polygon zeichnen. Bei jedem Eckpunkt des Polygons drücken Sie die linke Maustaste, dann wird der aktuelle Punkt mit dem letzten durch eine Linie verbunden. Dieses Polygon darf sich allerdings nicht selbst »schneiden«, das heißt, die Verbindungslinien der Eckpunkte dürfen sich nicht berühren. Wenn Sie genug Eckpunkte für das Polygon haben, drücken Sie die rechte Maustaste. »Construct« schließt automatisch das Polygon, indem es den ersten mit dem letzten Punkt verbindet.

Danach fragt »Construct«, ob Sie mit dem Ergebnis zufrieden sind; wenn nicht, können Sie die ganze Prozedur wiederholen. Wenn Sie zufrieden sind, fragt »Construct«, ob Sie »Löcher« einfügen wollen. Ein Loch ist hier lediglich ein weiteres Polygon, das vollständig im Inneren des ersten liegt. Die Löcher werden später beim Triangulieren ausgelassen. Die Löcher dürfen sich nicht mit dem ersten Polygon und auch nicht mit anderen Löchern schneiden.

Haben Sie genug Löcher eingegeben, können Sie entscheiden, ob das Polygon flach oder dreidimensional werden soll. Im letzteren Fall müssen Sie mit dem Proportionalregler die »Dicke« eingeben.

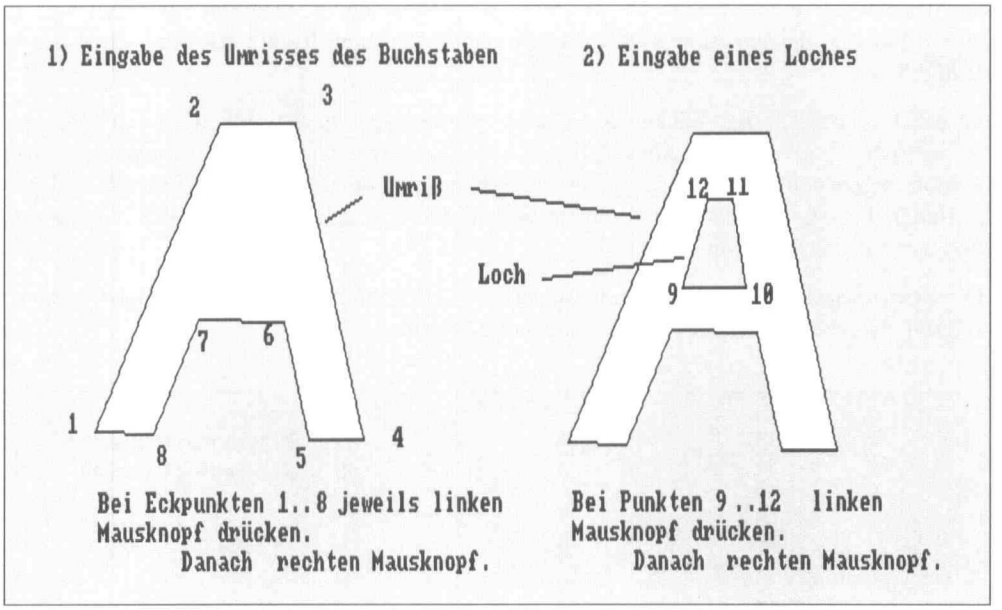


Bild 3.4: Erstellen eines Buchstabens

Danach trianguliert »Construct« das Polygon. Zuerst werden die Punkte mit Kanten verbunden, danach die Kanten zu Dreiecken zusammengefügt. Bei großer Eckpunktanzahl kann das Triangulieren schon ein paar Minuten dauern. Wundern Sie sich also nicht, wenn eine Weile nichts passiert; »Construct« rechnet dann konzentriert. Bild 3.4 zeigt die Eingabe eines Buchstabens mit 3D-Polygon.

Nachdem Sie das Polygon gezeichnet haben, können Sie es aus Richtung »vorne«, also von Süd nach Nord, betrachten.

Rotat.Körper

Diese Funktion dient zur Erzeugung von Rotationskörpern. Wir haben solche Körper schon kennengelernt. Kegel und Zylinder sind spezielle Rotationskörper. Mit diesem Menüpunkt können Sie aber die Form der Rotationskörper völlig frei bestimmen. Bei Anwahl von »Rotat.Körper« füllt sich das Plotfenster mit einem Gitter. In der Mitte des Plotfensters befindet sich eine waagrechte Linie; sie dient als Rotationsachse. Sie können jetzt einen Linienzug eingeben.

Nach Klicken der linken Maustaste wird der Punkt an der Position des Mauszeigers mit dem zuletzt bestimmten durch eine Linie verbunden. Bei Klicken mit der rechten Taste wird zwar ein neuer Punkt geschaffen, aber nicht mit dem letzten verbunden. Dies dient dazu, »Lücken« im Rotationskörper zu schaffen. Die Linien werden spiegelbildlich auch auf der jeweils anderen Seite der Rotationsachse gezeichnet. Dadurch sehen Sie immer eine Silhouette des Rotationskörpers. Ein Beispiel für einen Rotationskörper sehen Sie in Bild 3.5.

Ist der Linienzug fertig, klicken Sie das »Fertig«-Gadget in der linken oberen Ecke an. »Construct« fragt dann, ob Sie mit dem Linienzug zufrieden sind. Sie können die ganze Aktion gegebenenfalls wiederholen. Anschließend möchte »Construct« wie bei Kegel und Zylinder die Anzahl der Winkelschritte wissen. Danach geben Sie noch den Namen ein, und fertig ist der Rotationskörper.

Die Rotationsachse ist die Ost-West-Achse, das heißt, Sie haben Ihren Rotationskörper zu Beginn waagerecht liegend von West nach Ost orientiert.

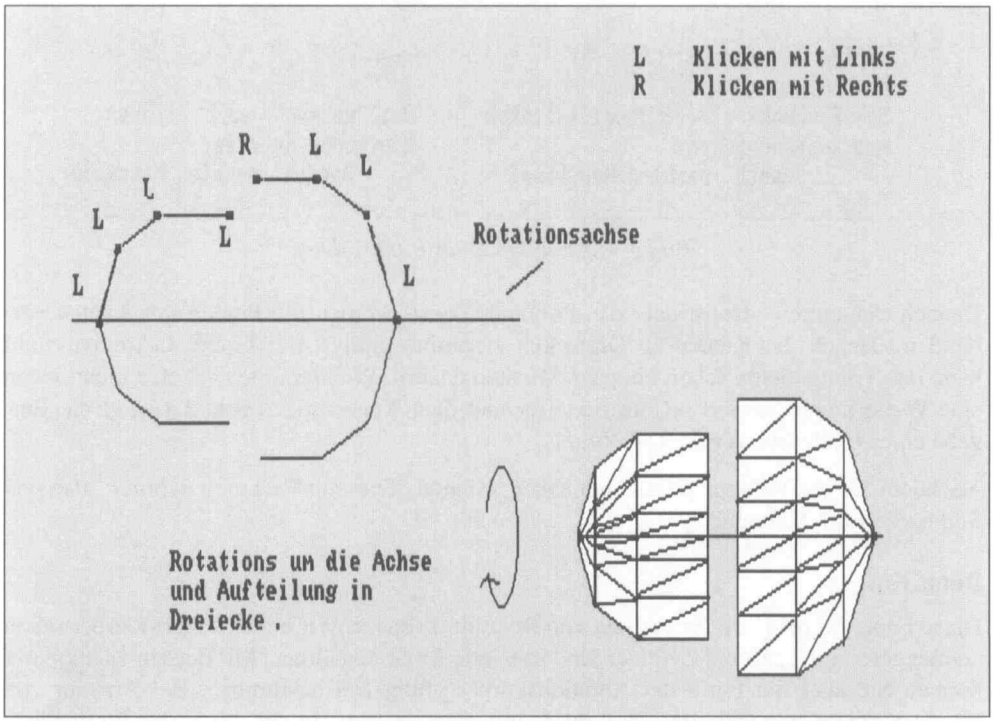


Bild 3.5: Beispiel für einen Rotationskörper

Schnitt.Krp

Stellen Sie sich einen Klumpen aus einem bestimmten Material vor, vielleicht aus Ton, Lehm oder Plastik. Jetzt schneiden Sie in Gedanken von dem Klumpen mit einem scharfen Messer von oben nach unten eine Scheibe ab. Danach die nächste Scheibe und immer weiter von vorne nach hinten, bis der Klumpen nur noch aus Scheiben besteht.

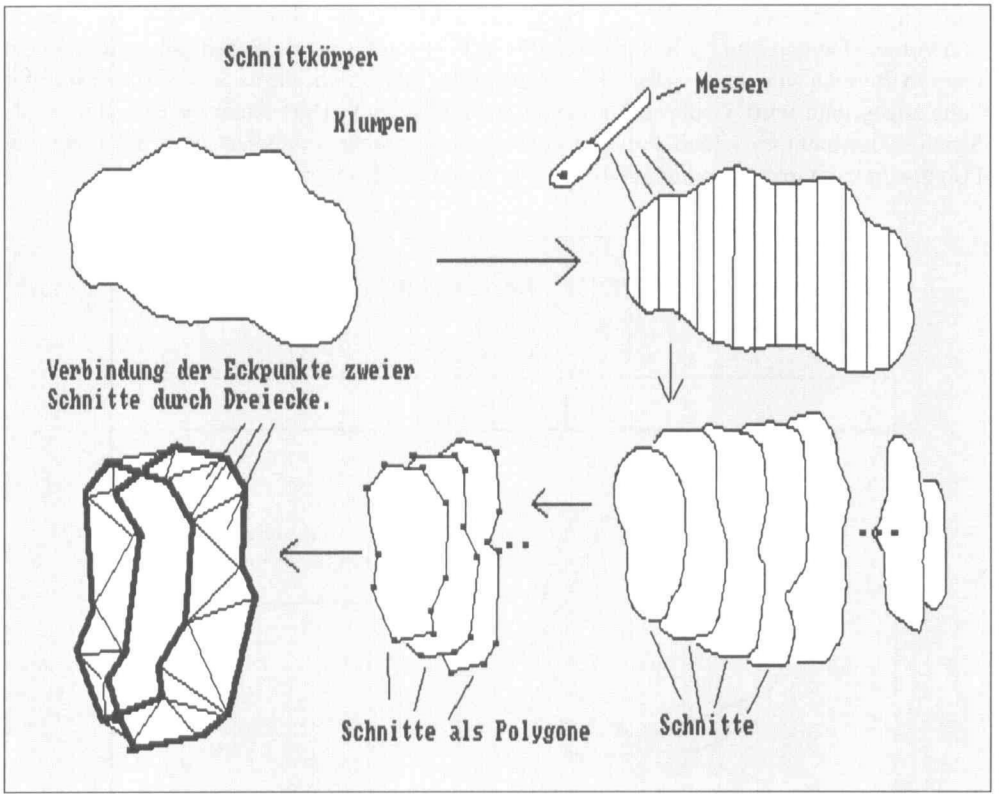


Bild 3.6: Schnittkörper

Legen Sie sich jetzt in Gedanken die Scheiben einzeln auf jeweils ein Blatt Papier und zeichnen Sie den Umriß mit einem Bleistift nach. Danach haben Sie einen Stapel von Papierblättern mit dem Umriß einer der Scheiben.

Eine solche Sammlung benötigt »Construct«. Bei Reflections können Sie solche Umrisse als Polygone annähernd genau mit der Maus eingeben, Scheibe für Scheibe. Danach verbindet »Construct« die Eckpunkte benachbarter Vielecke (also Scheiben) durch Dreiecke. Es überzieht somit die Schnitte mit einer Haut aus Dreiecken und erzeugt dadurch einen Körper, der ungefähr die Form des ursprünglichen Klumpens hat.

Ich hoffe, das war verständlich genug. Sie können natürlich auf eine Vorbereitung mit Klumpen, Messer und Papier verzichten und die Umrisse der Scheiben gleich frei aus dem Gedächtnis und nach Ihrer Phantasie zeichnen.

Nun zur Handhabung von »Schnitt.Krp«: Bei Anwahl von »Schnitt.Krp« füllt sich das Plotfenster, wie in Bild 3.7 gezeigt. Das Fenster teilt sich in verschiedene Bereiche: Der große Bereich dient zur Eingabe der Schnitte. Sie geben dort die Schnitte - durch Polygone angenähert - ein (wie bei der Funktion »3D-Polygon«). Hier sehen Sie die Schnitte von vorne, also von Süd nach Nord. Sobald ein Polygon fertig ist, können Sie es im rechten unteren Bereich von links sehen. Es erscheint nur als Strich, da es ja senkrecht von der Seite angeschaut wird. Genauso im rechten oberen Bereich. Dort sehen Sie es von oben als Strich. Mit einem einzelnen Polygon können Sie da nichts erkennen, aber mit mehreren Polygonen kann man schon ungefähr die Silhouette des Körpers abschätzen.

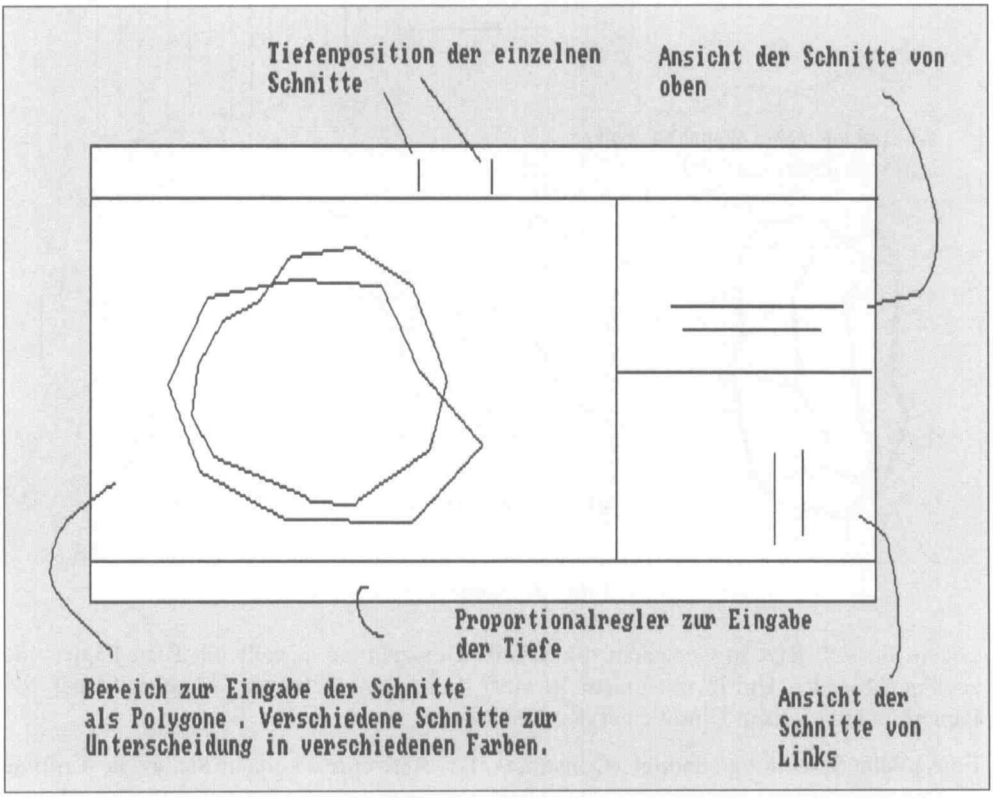


Bild 3.7: Schnitte

Ist ein Polygon fertig, fragt »Construct«, ob Sie noch mehr Schnitte wünschen. Wenn ja, müssen Sie ab jetzt vor jedem Schnitt die Tiefe des Schnittes im Klumpen eingeben. Dies geschieht über den Proportionalregler. An der entsprechenden Stelle im oberen Bereich erscheint wieder eine senkrechte Markierung, die der Tiefenposition entspricht. Danach geben Sie das nächste Polygon ein. Bei jedem Schnitt wechselt »Construct« übrigens die Farbe, und wenn ihm die Farben ausgehen, die Strichart, so daß Sie die einzelnen Schnitte einigermaßen unterscheiden können. Die Tiefenmarkierungen entsprechen in der Farbe ihren Schnitten. Die Reihenfolge der Schnitte ist egal. »Construct« sortiert sie am Ende nach der Tiefe.

Wenn Sie mit allen Schnitten fertig sind, verbindet »Construct« die Polygone durch Dreiecke und schließt außerdem das oberste und unterste Polygon durch Triangulierung. Zum Schluß muß nur noch der Name angegeben werden.

Der Körper ist jetzt so orientiert, daß die Schnitte in der Ost-West-Oben-Unten-Ebene liegen. Die Tiefe ist hier die Süd-Nord-Richtung. Sobald der Körper fertig ist, zeichnet ihn »Construct« automatisch von links. (Haben Sie Geduld mit dieser Funktion. Es erfordert einiges an Übung, um den Körper so zu gestalten, wie man es sich vorgestellt hat. Aber: Übung macht den Meister.)

3.3 Darstellungsarten

In diesem Kapitel werden die verschiedenen Möglichkeiten erklärt, sich die Körper, die man erzeugt hat, anzuschauen. Dies ist nicht nur wichtig, um sich einen Überblick über die Szene verschaffen zu können, sondern die Darstellungsarten beeinflussen auch die diversen Funktionen zur Modifikation von Körpern. Sie können folgende Parameter beim Plotten beeinflussen:

Blickrichtung: Von welcher Seite aus wird ein Körper dargestellt?

PlotModus: Wie werden die Körper dargestellt?

FarbModus: Wie werden die Objekte eines Körpers farblich dargestellt?

Bei der Blickrichtung unterscheidet »Construct« zwischen perspektivischer und achsenparalleler Darstellung. Mit der perspektivischen Darstellung können Sie sich einen Körper aus beliebiger Richtung anschauen. Die achsenparallele Darstellung wiederum dient vor allem zur Manipulation von Körpern, weil alle Operationen, die Sie mit Körpern ausführen, nur achsenparallel stattfinden. Alle Parameter werden im »Plot«-Menü als Untermenüpunkte zusammengefaßt. Die Funktionen können auch über Tasten aufgerufen werden, da sie sehr häufig benutzt werden. Die entsprechenden Tastenfunktionen stehen hinter den Funktionen. [RAmiga]+[k] bedeutet z.B.: gleichzeitiges Drücken der rechten Amiga-Taste (neben der langen Leerzeichen-Taste) und der [k]-Taste.

[RAmiga]+[k] **PlotKörp**

Hier wird der aktuelle Körper bestimmt, der dargestellt werden soll. Alle folgenden Plot-Befehle beziehen sich auf diesen Körper, bis Sie einen neuen bestimmen. Dieser Körper wird im folgenden »Plotkörper« genannt. Anfangs ist als »Plotkörper« der Körper »all« eingestellt.

[RAmiga]+[f] **Fenster**

Diese Funktion entspricht der »Plotkörp«-Funktion, nur wird zusätzlich das Plotfenster auf den Körper eingestellt. »Construct« berechnet den Umfang des Körpers. Dies wird bei den Funktionen »Vorne«, »Hinten«, »Links«, »Rechts«, »Oben« und »Unten« gebraucht, damit das Plotfenster so auf den Körper eingestellt wird, daß er vollständig im Fenster erscheint. Bei einem Körper, der aus Tausenden von Dreiecken besteht, kann das schon einige Sekunden dauern.

Wenn Sie sich einen Körper genau anschauen wollen, können Sie mit »Fenster« den Maßstab auf den Körper einstellen.

[RAmiga]+[v] **Vorne**

Mit diesem Punkt wird sowohl die Darstellungsart als auch die Blickrichtung eingestellt. Der Körper wird dabei parallel auf den Bildschirm projeziert, das heißt, es erfolgt keine perspektivische Darstellung.

Die Parallelprojektion ist für viele Operationen wichtig. Alle geometrischen Operationen werden in irgendeiner Weise parallel zu den Koordinatenachsen ausgeführt. Das heißt, wenn Sie eine dieser Operationen (siehe Abschnitt 3.5) ausführen wollen, müssen Sie den gewünschten Körper aus einer der Richtungen »Vorne«, »Hinten«, »Links«, »Rechts«, »Oben« oder »Unten« (s.u.) anschauen.

Bei »Vorne« schauen Sie den Körper von Süden nach Norden an. Bevor der Körper gezeichnet wird, wird das Plotfenster gelöscht.

[RAmiga]+[h] **Hinten**

Wie »Vorne«, nur Blickrichtung von Norden nach Süden.

[RAmiga]+[l] **Links**

Wie »Vorne«. Sie schauen den Körper von links aus an. Blickrichtung von West nach Ost.

[RAmiga]+[r] **Rechts**

Wie »Links«, nur Blickrichtung von Ost nach West.

[RAmiga]+[o] **Oben**

Sie schauen den Körper von oben aus an. Blickrichtung ist von oben nach unten.

RAmiga+u **Unten**

Wie »Oben«, nur Blickrichtung von unten nach oben.

RAmiga+p **Persp.**

Hier verlassen wir die parallele und kommen zur perspektivischen Projektion. Der Körper wird jetzt nicht mehr aus einer festen Richtung, entlang der Koordinatenachsen, angeschaut, sondern von einem frei wählbaren Augenpunkt aus in beliebiger Vergrößerung.

Augenpunkt, Bildebene, Blickrichtung, Vergrößerung und einige weitere Parameter werden durch die Funktionen »Kamera« und »Kam.beweg« bestimmt. Die dort eingestellten Parameter werden durch »Persp.« benutzt (s.u.). »Persp.« funktioniert ansonsten genauso wie »Vorne«, .., »Unten«. Sie müssen allerdings selbst dafür sorgen, daß Sie auf den richtigen Körper zielen. Wenn Sie mit »Kamera« und »Kam.beweg« auf einen Körper x gezielt haben und sich dann mit »Persp« den Körper y anschauen wollen, der aber ganz woanders als x liegt, bleibt das Plotfenster leer.

Die geometrische Operationen lassen sich bei perspektivischer Darstellung nicht ausführen. »Construct« verweigert hier solche Operationen.

Bei »Persp.« erscheint im Plotfenster rechts und links je eine gepunktete Linie. Sie zeigen an, wo der Bildrand beim Raytracing ist.

RAmiga+c **Plotten**

Der zuletzt eingestellte Plotkörper (mit »Plotkoerp« oder »Fenster«) wird mit der zuletzt gewählten Darstellungsart geplottet. Das Plotfenster wird zuvor nicht gelöscht.

Die folgenden Funktionen beeinflussen die Plot- sowie die Farbmodi.

Hidd_0

Im »Hidd_0«-Modus zeichnet »Construct« alle Objekte eines Körpers. Dies mag selbstverständlich erscheinen, aber bei umfangreichen Körpern wird oftmals das Plotfenster dermaßen mit Linien »zugezeichnet«, daß man kaum noch etwas erkennen kann. Hierfür gibt es die beiden folgenden Plotmodi:

Hidd_1

Bei »Hidd_1« zeichnet »Construct« nur diejenigen Dreiecke, die dem Betrachter zugewandt sind. Die Dreiecke an der Rückseite des Körpers (vom Betrachter aus) werden weggelassen. Diese Maßname erhöht die Übersichtlichkeit schon sehr, aber Sie sehen nicht alle Dreiecke. Kugeln sind übrigens von diesen Plot-Modi nicht betroffen; sie werden immer gezeichnet. Ihre Oberfläche zeigt ja in alle Richtungen.

»Hidd_1« ist kein echtes »Hidden_line«. Es werden keine Verdeckungen berücksichtigt, sondern nur aufgrund der Richtung eines Dreieckes errechnet, ob es vom Betrachter überhaupt zu sehen ist.

OK — Wenn Sie mit der Position zufrieden sind, klicken Sie OK an. Danach verschwinden die Felder und der Quader, statt dessen wird der richtige Körper geplottet.

bewegen — Hiermit können Sie direkt zur Funktion »Kam.beweg« springen (siehe unten).

Abbruch — Hiermit wird die Position, die die Kamera vor Beginn der Funktion »Kamera« hatte, wiederhergestellt.

Das Verfahren, den Plotkörper durch einen Quader zu umschließen, mag vielleicht etwas befremden, »Construct« könnte doch gleich den Plotkörper zeichnen und entsprechend drehen, werden Sie sich sagen. Bei Körpern, die nur aus wenigen Objekten bestehen, würde das sicherlich funktionieren, nur Körper können eben auch sehr umfangreich an Objekten sein. Bei einem Körper mit etwa 1000 Objekten dauert der Bildaufbau schon ein paar Sekunden. Von einer schnellen und bequemen Bestimmung der Kameraposition könnte keine Rede mehr sein.

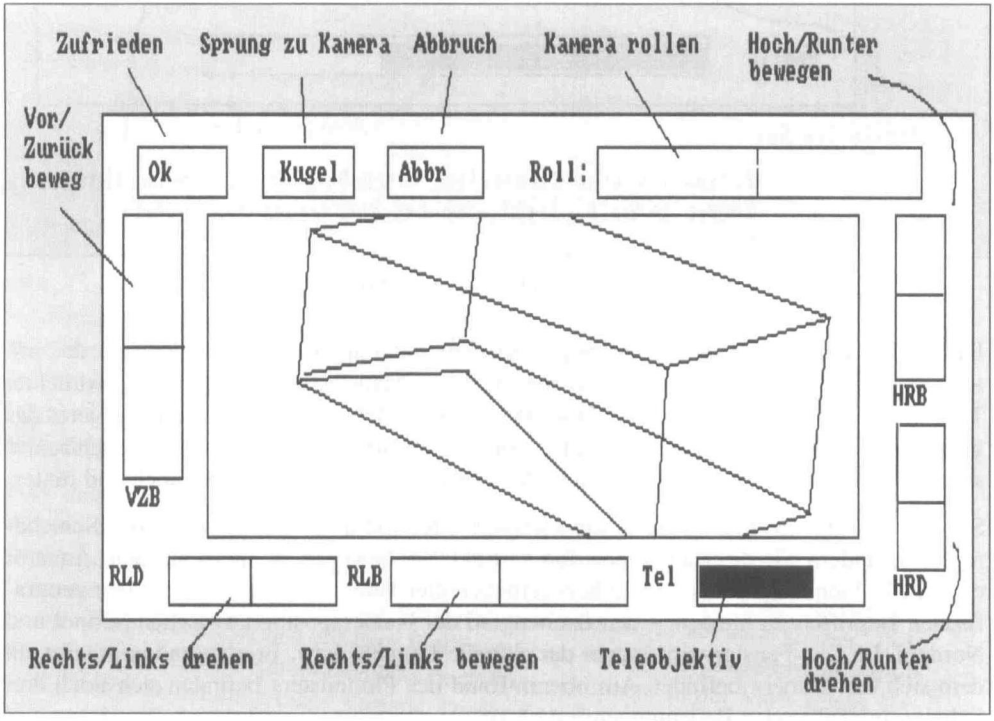


Bild 3.9: Die Regler zur Kamerabewegung

Kam.beweg
Mit der Funktion »Kamera« können Sie die Position der Kamera frei auf einer gedachten Kugeloberfläche, mit dem Plotkörper im Zentrum, bestimmen. Mit »Kam.beweg« können Sie jetzt von dieser Oberfläche abweichen und die Kamera frei bewegen.

Wie bei »Kamera« erscheinen im Plotfenster wieder einige Felder und der Kasten, der den minimal umschließenden Quader des Plotkörpers darstellt. Allerdings haben Sie jetzt einige Schieberegler mehr zur Verfügung. Sie finden Regler zur Bewegung der Kamera, zur Drehung der Kamera sowie einen zur Einstellung des Teleobjektives. Die Regler werden genauso wie bei »Kamera« bedient, nämlich durch Anklicken mit der linken Maustaste und Bewegung der Maus bei gedrückter Taste (Bild 3.9).

Betrachten wir die Regler der Reihe nach:

Links ist ein senkrechter Regler, um die Kamera in Blickrichtung zu bewegen.

VZB: Dies steht für Vor-Zurück-Bewegung. Sie können hiermit die Kamera vor- und zurückbewegen. Oberhalb der Mittelmarkierung bedeutet Bewegung nach vorne, unterhalb nach hinten.

Die beiden Regler unten links dienen zur Bewegung der Kamera in Rechts-Links-Richtung:

RLB: Rechts-Links-Bewegen. Durch Verändern des Reglers verschieben Sie die Kamera nach rechts oder links. Die Blickrichtung bleibt unverändert.

RLD: Das »D« am Ende steht für drehen. Statt sich zu bewegen, dreht sich die Kamera hier nur nach rechts bzw. links. Die Bewegung des Quaders am Fenster sieht ähnlich aus wie bei »R-L-B«, aber hier wird die Blickrichtung der Kamera verändert und bei »R-L-B« die Position. Daneben ist das Teleobjektiv.

Tel: Durch Linksschieben des Reglers können Sie sich den Körper nahe heranholen bzw. vergrößern. Umgekehrt kann er auch verkleinert werden.

Eine Vergrößerung wie bei »Tele« können Sie auch erreichen, indem Sie mit »V-Z-B« auf den Körper zufahren. Kommen Sie aber zu nahe heran, wird der Körper stark verzerrt.

Auf der rechten Seite sind noch Regler zur Auf/Ab-Bewegung der Kamera.

HRB: Hoch-Runter-Bewegen. Wie bei »R-L-B«.

HRD: Wie bei »R-L-D«.

Es ist noch zu beachten, daß sich die Bewegungen ausschließlich auf die Orientierung der Kamera beziehen. Bei Vor/Zurück-Bewegung bewegt sich die Kamera in Blickrichtung oder dagegen. Bei Rechts/Links und Hoch/Runter bewegt sie sich relativ zur Blickrichtung und nicht nach den Koordinatenachsen.

Oben rechts ist schließlich noch ein Regler, um den Rollwinkel zu beeinflussen.

Roll: Hiermit können Sie die Kamera um die Blickrichtung rollen lassen.

Der Rollwinkel wird durch den Tele-Regler aber auf 0 zurückgestellt. Rollen Sie also erst, nachdem Sie die Vergrößerung eingestellt haben.

Die oberen Felder entsprechen denen bei »Kamera«:

OK Wenn Sie mit der Kameraposition zufrieden sind, klicken Sie »OK« an. Danach wird der Plotkörper gezeichnet.

Kugel Hiermit kommen Sie wieder zur oben beschriebenen Funktion »Kamera«. Damit verlieren Sie allerdings die jetzt eingestellte Kameraposition.

Abbruch Abbruch des ganzen, danach hat die Kamera wieder die gleiche Position wie vor Beginn von »Kam.bewg«.

3.4 Körper - Bausteine einer Szene

Im Überblick wurden bereits die Körper angesprochen. Hier sollen sie genauer vorgestellt werden: Ein Körper ist eine beliebige Teilmenge aller Objekte, die im Augenblick der Definition des Körpers vorhanden sind, verbunden mit einem Namen, unter dem er angesprochen werden kann. Intern gehört zu jedem Körper eine Liste mit Objektnummern. Anhand dieser Liste kann »Construct« alle Objekte bestimmen, die von einer Operation betroffen sind, die an einem Körper vorgenommen werden soll.

Wie werden Körper definiert? Keine Angst, mit Objektnummern haben sie dabei nichts zu tun. Folgende Möglichkeiten sind zur Objektdefinition gegeben:

- Ein Körper ist automatisch definiert. Er heißt »all« und wird vom System automatisch erzeugt und korrigiert. Wie der Name schon sagt, kann man mit ihm alle Objekte auf einmal ansprechen.
- Im »Tools«-Menü sind alle Möglichkeiten zur Erzeugung von Körpern zusammengefaßt. Jedesmal, nachdem Sie damit einen Körper erzeugt haben, fragt »Construct« nach dem Namen für den Körper. Unter diesem Namen können Sie ihn jetzt immer ansprechen.
- Im »Körper«-Menü sind alle Möglichkeiten zur Erzeugung und Manipulation von Körperbeschreibungen zusammengefaßt.

[RAmiga]+[z] **Zeigen**

Mit »Zeigen« können Sie sich einige Informationen über einen Körper verschaffen. Diese werden im Listfenster aufgelistet. Gezeigt wird die Anzahl Objekte, die im Körper enthalten sind, sowie alle vorkommenden Materialien in der Reihenfolge, in der Sie im Körper vorkommen. Diese Angaben erscheinen im Listfenster. Interessant ist diese Funktion in Verbindung mit dem Farbmodus »Farb_2« (siehe 3.3). Dann werden die Materialnamen

nämlich jeweils in der gleichen Farbe aufgelistet, die die entsprechenden Objekte beim Plotten haben. (Sie können sich einen Körper im Plotfenster und darüber im Listfenster die zu den Farben gehörenden Materialien anschauen.)

[RAmiga]+[+] **AddNeu:**

Mit »AddNeu« werden mehrere Körper zu einem neuen zusammengefaßt, die Körperbeschreibungen werden addiert. Dieser neue ist dann unter einem eigenen Namen ansprechbar. Es werden keine neuen Objekte erzeugt, sondern nur eine neue Körperbeschreibung angelegt. Bei Anwahl von »AddNeu« müssen Sie im Auswahlfenster die Körper anklicken (mit der linken Maustaste), die Sie zusammenfassen wollen. Wenn Sie fertig sind, klicken Sie entweder ein leeres Feld an, oder Sie drücken die rechte Maustaste. Danach muß nur noch der Name des neuen Körpers angegeben werden.

Beispiel: Sie haben sich schon die Körper »Sofa«, »Sessel1« und »Sessel2« geschaffen, geeignet positioniert und wollen diese drei Körper nun irgendwo anders in der Szene plazieren. Anstatt nun alle drei Körper einzeln zu bewegen, fassen Sie die Körper mit »AddNeu« zusammen als neuen Körper »Sitzgruppe«. Bei der Bewegung von »Sitzgruppe« werden »Sofa«, »Sessel1« und »Sessel2« automatisch mitbewegt.

Sie können beliebige Körper zusammenfassen. »Sitzgruppe« kann zum Beispiel aus »Tisch«, »Stuhl« und »Wänden« zu »Zimmer« zusammengefaßt werden.

[RAmiga]+[*] **Add**

»Add« entspricht im Prinzip »AddNeu«, nur wird hierbei kein neuer Körper geschaffen. Statt dessen wählen Sie zuerst einen Körper, der erweitert werden soll, und danach die Körper, die zu dem ersten dazugezählt werden. Damit können Sie nach und nach einen Körper um andere Körper erweitern.

[RAmiga]+ [-] **Sub**

Mit »Sub« werden Körper voneinander »abgezogen«. Wieder ist dies nur eine Angelegenheit der Körperverwaltung. Objekte werden davon nicht berührt. Bei Anwahl von »Sub« müssen Sie zuerst den Körper anklicken, von dem etwas abgezogen werden soll. Danach klicken Sie die Körper an, die Sie von dem ersten abziehen wollen. Am Ende drücken Sie wieder die rechte Maustaste. Mit »Sub« könnten Sie beispielsweise »Sessel2« wieder von »Sitzgruppe« abziehen.

[RAmiga]+[d] **Define**

»Define« dient zum freien Definieren von Körpern, und zwar auf grafische Weise. Sie teilen »Construct« mit, daß alle Objekte, die sich innerhalb eines Rechtecks befinden, nun als ein Körper zu betrachten sind.

Sobald Sie »Define« anklicken, müssen Sie »Construct« mitteilen, aus welchem Körper Sie mit »Define« einen Teil herauspicken wollen. Dies geschieht – wie üblich – durch An-

klicken des Körpernamens im Auswahlfenster. Danach fragt »Construct« nach dem Auswahlmodus. Es kennt zwei:

- Modus0: Es werden alle Dreiecke genommen, die sich mit allen drei Punkten innerhalb des Rechtecks befinden.
- Modus1: Es werden alle Dreiecke genommen, von denen sich mindestens ein Eckpunkt im Rechteck befindet.

Bild 3.10 verdeutlicht die beiden Möglichkeiten.

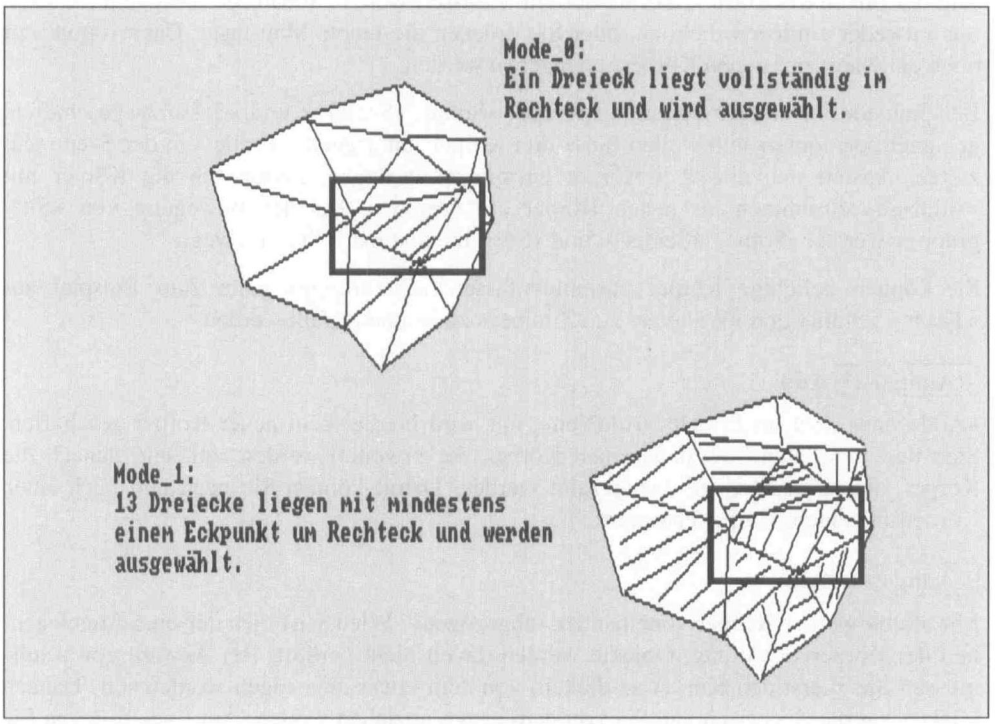


Bild 3.10: Die beiden Modi Mode_0 und Mode_1

Kugeln werden unabhängig vom Modus immer genommen, wenn sich der Mittelpunkt innerhalb des Rechtecks befindet. Es erscheint dann das (oben beschriebene) Rechteck. Das müssen Sie über die Objekte legen, die Sie auswählen wollen. Danach zeichnet »Construct« alle Objekte, die es im gewählten Modus innerhalb des Rechtecks gefunden hat, in weißer Farbe. Diesen Schritt können Sie mehrmals wiederholen, bis Sie alle Objekte beisammen haben, die den neuen Körper bilden sollen. Am Ende will »Construct« nur noch den Namen des neuen Körpers wissen.

Es ist noch wichtig, daß »Define« eng mit dem Plotmodus zusammenhängt. Es werden nämlich nur die Objekte gewählt, die tatsächlich zu sehen sind. Ist der Körper mit »Hidd_0« gezeichnet, dann werden mit »Define« nur Objekte an der dem Betrachter zugewandten Seite des Körpers ausgewählt. Wollen Sie auch die Objekte an der Rückseite, müssen Sie zuvor den Körper mit »Hidd_0« oder »Hidd_2« zeichnen.

Doppeln

Dieser Untermenüpunkt erklärt sich schon durch seinen Namen. Sie können hiermit einen Körper verdoppeln. In diesem Fall wird nicht nur eine neue Körperbeschreibung angelegt, wie bei den Punkten »Add«, »Sub« und »Define«, sondern es werden neue Objekte und Punkte erzeugt.

Klicken Sie im Auswahlfenster den Körper an, den Sie verdoppeln wollen. »Construct« fragt Sie dann nach dem Namen, den der Doppelgänger des angewählten Körpers haben soll. Danach wird von jedem Objekt und jedem Punkt des Körpers eine Kopie angelegt. Der neue Körper enthält dann eine Beschreibung der neuerzeugten Objekte.

Doppeln ist ein sehr wichtiges Hilfsmittel, um schnell eine komplexe Szene zu schaffen, doch sind dabei auch einige Punkte zu beachten:

- Das Duplikat befindet sich genau an der gleichen Stelle wie das Original. Sie werden es also noch nicht sehen können. Dazu müssen sie eins von beiden entweder verschieben, drehen oder sonstwie verändern.
- Das Duplikat enthält genau die gleichen Materialeigenschaften wie das Original.
- Es werden nur Objekte und Punkte verdoppelt, keine Körper. Dies sollte näher erläutert werden: Nehmen wir wieder unser obiges Beispiel mit dem Körper »Sitzgruppe«. Wir verdoppeln »Sitzgruppe« zu »Gruppe2«.

  Damit haben wir zwei Körper, die jeweils eine Sitzgruppe beschreiben. Wir können »Gruppe2« in eine andere Ecke des Zimmers schieben. Erinnern wir uns, daß »Sitzgruppe« aus einem Sofa und zwei Sesseln besteht (in Wirklichkeit natürlich nur aus den Dreiecken, die diese Gegenstände bilden). Da »Gruppe2« ein Duplikat von »Sitzgruppe« ist, muß es da auch ein Sofa und zwei Sessel geben. Jetzt kommt der Problempunkt: Für die Sessel und das Sofa aus »Gruppe2« gibt es keine Körperbeschreibungen. Wir können damit vorerst auf diese Gegenstände aus »Gruppe2« nicht zugreifen und sie individuell bewegen. Es ist aber kein Problem, sich entsprechende Körperbeschreibungen zu erzeugen:

  Plotten Sie »Gruppe2« auf dem Plotschirm, allerdings mit Plotmodus »Hidd_0« oder »Hidd_2«. Wählen Sie Untermenü »Define« und schieben Sie das Rechteck zum Beispiel über einen der Sessel. Mit »Define« können Sie sich eine Körperbeschreibung für jeden beliebigen Gegenstand Ihrer Szene anlegen.
- Wenn ein Verdoppelungsschritt die maximale Anzahl Punkte und Objekte überschreiten würde, wird er selbstverständlich nicht ausgeführt.

Material
Hiermit können Sie einem Körper eine Materialeigenschaft geben. Nach Anwahl von »Material« klicken Sie im Auswahlfenster den Körper an, dem Sie eine Materialeigenschaft verpassen wollen. Sodann erscheint im Auswahlfenster die Liste aller Materialien, die »Construct« zur Zeit kennt. Alle Objekte des Körpers erhalten die angeklickte Materialeigenschaft.

Mit »Material« kommen wir dem Raytracing schon etwas näher, denn das braucht ja Materialeigenschaften. Nur in »Construct« haben wir von all den schönen Materialien wie Spiegel, Glas, Marmor etc. nichts, denn hier werden die Objekte nur als Strichgrafik dargestellt. »Construct« kann nur verschiedene Materialien durch verschiedene Strichfarben deutlich machen. Folgendes ist wichtig:

Alle Objekte des Körpers erhalten die gleiche Materialeigenschaft. Geben Sie »Sitzgruppe« die Eigenschaft »Schwarzes_Leder«, dann erhalten die Objekte von »Sofa«, »Sessel1« und »Sessel2« auch diese Eigenschaft, da sie Teil von »Sitzgruppe« sind. Wenn der Sessel vorher eine andere Eigenschaft hatte, ist das alte Material jetzt »überschrieben«. »Material« ist also eine Funktion, bei der man auch sehr viel falsch machen kann. Man muß sich genau überlegen, welchem Körper man welche Eigenschaft zuordnet und welche anderen Körper eventuell davon beeinflußt werden.

Hier pfuschen uns aber die Texturen ins Handwerk. Mit ihnen hat es etwas Besonderes auf sich:

Manche Materialien beinhalten Texturmuster, die im Auswahlfenster farblich anders gekennzeichnet sind. Bevor Sie einem Körper ein Textur-Material geben, lesen Sie bitte vorher Abschnitt 3.7.

K_löschen
Mit der Zeit sammeln sich sehr viele Körperbeschreibungen an. Abgesehen davon, daß sie Speicher verbrauchen, wird es mit der Zeit ziemlich unübersichtlich, wenn man mit sehr vielen Körpern arbeitet. Mit »K_löschen« können Sie Körper entfernen. Nach Anwahl von »K_löschen« klicken Sie im Auswahlfenster die Körper an, die Sie gelöscht haben wollen. Wollen Sie keinen weiteren löschen, drücken Sie die rechte Maustaste.

O_löschen
»O_löschen« ist ähnlich wie »K_löschen«, nur werden zusätzlich die Objekte gelöscht. Überall, wo etwas gelöscht wird, ist Vorsicht geboten. Nachdem Sie einen Körper angeklickt haben, den Sie löschen wollen, fragt »Construct« nach, ob Sie sich auch sicher sind. Wenn ja, werden alle Objekte, die der Körper enthält, gelöscht (Einschränkungen: siehe Abschnitt 3.7 (Textur)).

»Construct« hat daraufhin einiges an Verwaltungsarbeit zu tun:

Objekte, die im Speicher hinter den gelöschten liegen, müssen aufgeschoben werden. Dadurch verändern sich Ihre Objektnummern. Alle Körperbeschreibungen müssen demnach korrigiert werden. Außerdem können Ihre gelöschten Objekte auch Teil anderer Körper gewesen sein. Wenn Sie »Sofa« löschen, muß auch »Sitzgruppe« korrigiert werden, da die gelöschten Objekte auch dessen Teil waren. Löschen Sie andererseits »Sitzgruppe«, müssen die Körper »Sessel1«, »Sessel2« und »Sofa« ebenso gelöscht werden, da ihre Objekte jetzt nicht mehr existieren.

Schneiden

Schneiden ist eine spezielle Funktion. Sie können mit ihr einen Körper aus einem anderen herausschneiden. Sie meinen vielleicht, das könnte man schon mit »Sub« erreichen? Nein, mit »Schneiden« hat es noch etwas mehr auf sich:

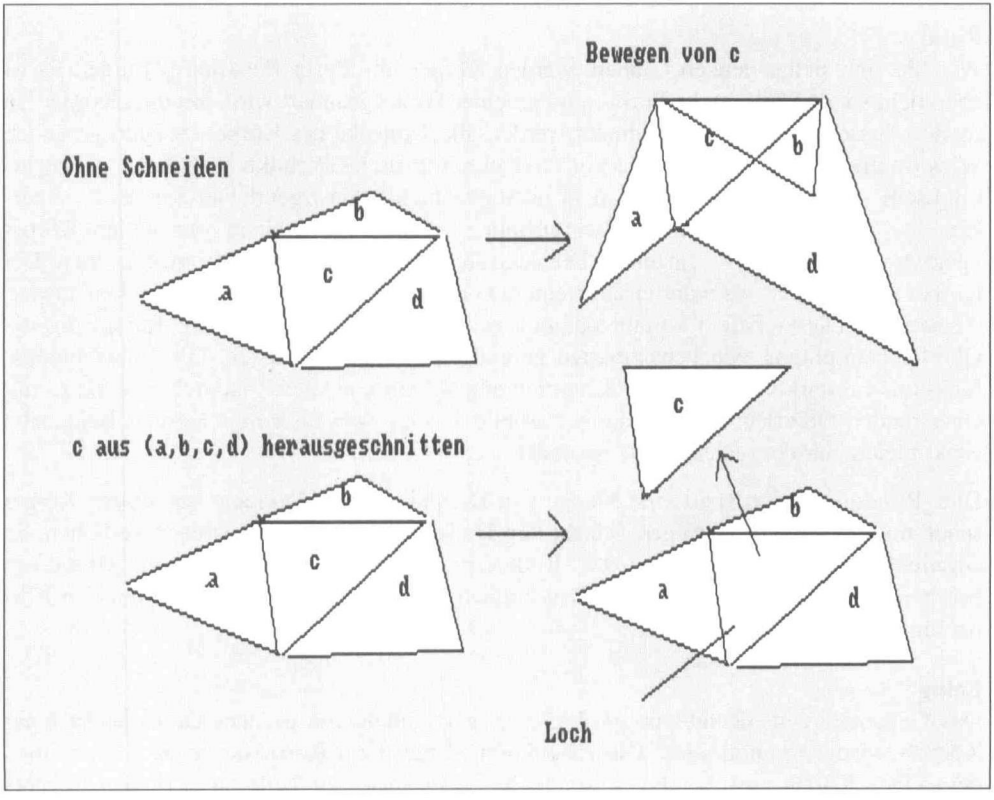


Bild 3.11: Verschieben eines Objektes, mit und ohne Schneiden

Dreiecke enthalten drei Zeiger auf ihre drei Eckpunkte. Benachbarte Dreiecke zeigen teilweise auf gleiche Eckpunkte, das heißt, von verschiedenen Dreiecken aus wird oft auf die gleichen Punkte gezeigt. Bewegt man nun ein Dreieck, das Punkt x als Eckpunkt enthält, so werden alle Dreiecke, die auch x enthalten, zumindest verzerrt. Insofern besteht zwischen benachbarten Dreiecken ein enger Zusammenhang, auch wenn Sie zu verschiedenen Körpern gehören. Die Punkte »schweißen« die Dreiecke sozusagen zusammen.

»Schneiden« löst genau diesen Zusammenhang auf. Wenn Sie einen Körper x schneiden wollen, wird für jeden Punkt, auf den Objekte aus x zeigen und auf den auch Objekte von anderen Körpern zeigen, ein neuer Punkt an der gleichen Stelle angelegt. Die Objekte aus x werden so verändert, daß sie auf die neuen Punkte zeigen. Es ist damit jeder Zusammenhang von Objekten aus x mit anderen Objekten gelöst. Sie können jetzt x fortbewegen; da, wo es vorher war, bleibt ein Loch. Bild 3.11 verdeutlicht das Verschieben eines Objektes; einmal ohne »Schneiden«, einmal, nachdem es geschnitten wurde. Bei Anwahl von »Schneiden« müssen Sie lediglich den Körper angeben, der geschnitten werden soll.

Rund

Wie Sie sich sicher denken können, werden Körper mit dieser Funktion gerundet. Es ist aber nicht so, daß hierbei die Form in irgendeiner Weise geändert wird, um dem Körper ein rundes Aussehen zu geben. Vielmehr werden alle Dreiecke des Körpers pseudo-gerundet. Wieso pseudo? Es sind nach wie vor Dreiecke, mit drei Eckpunkten, da kann gar nichts rund sein. Außerdem werden Sie in »Construct« nichts von irgendeiner Rundheit bemerken. Die Rundheit wird Ihnen aber auffallen, nachdem Sie einen gerundeten Körper »geraytraced« haben. Die einzelnen Dreiecke sind so gut wie nicht mehr zu erkennen. Der Körper sieht so aus, als wäre er aus »einem Guß« und nicht aus vielen Dreiecken zusammengesetzt. Dieser Effekt kommt dadurch zustande, daß »Beams« beim Raytracing die Oberflächenrichtung von benachbarten gerundeten Dreiecken mischt. Die Beleuchtungsfunktion, die stark von der Oberflächenrichtung abhängt, »glaubt« dadurch, daß sie es mit einer runden Oberfläche zu tun hätte. Farbbild 1 zeigt zwei eiförmige Körper. Beide sind exakt gleich, nur der rechte wurde gerundet.

Die »Rund«-Funktion spart eine Menge von Dreiecken ein. Man kann nun einem Körper schon mit einer relativ geringen Anzahl von Dreiecken ein rundes Aussehen verleihen. Es empfiehlt sich, nach Anwendung von »Rotkoerp« dem entstandenen Körper mit »Rund« zu bearbeiten. Die Handhabung ist denkbar einfach. Sie klicken im Auswahlfenster den Körper an, der gerundet werden soll.

Eckig

Das Gegenteil von »Rund« ist »Eckig«. Es verwandelt alle gerundeten Dreiecke eines Körpers wieder in »normale«. Diese Funktion ist meist zur Korrektur gedacht, wenn man etwa einen Körper versehentlich gerundet hat, aber auch, um Teile eines runden Körpers wieder eckig zu machen. Es sollte noch erwähnt werden, daß alle Dreiecke, die neu erzeugt werden, immer eckig sind. Wenn Sie also runde Körper haben wollen, müssen Sie dies mit »Rund« selbst vornehmen.

Umdreh
Hiermit wird die Orientierung der Dreiecke eines Körpers umgedreht. Wenn Sie einen Körper umdrehen und danach mit »Hidd_1« plotten, sehen Sie nicht die Dreiecke der Ihnen zugewandten Seite, sondern die der abgewandten. Diese Funktion werden Sie normalerweise nicht benutzen. Sie ist als Hilfsmittel für Szenen gedacht, die mit dem Hilfsprogramm »get_sculpt« (siehe Kapitel 6) eingelesen wurden, da es dort Probleme mit der Orientierung gibt.

Def_Mat
»Def_Mat« ist wieder eine Funktion, um einen Teil eines Körpers als neuen Körper zu definieren. Bei »Define« wurde dies grafisch realisiert; dort wurden alle Objekte ausgewählt, die innerhalb eines Rechtecks liegen. Bei »Def_Mat« hingegen werden die Objekte nach ihrer Materialeigenschaft ausgewählt. Dies läßt sich natürlich nur machen, wenn bereits Materialeigenschaften vorliegen. Bei Anwahl von »Def_Mat« will »Construct« zuerst den Körper wissen, aus dem die Objekte ausgewählt werden sollen, und dann das Material, das die gesuchten Objekte haben. Anschließend sucht »Construct« im Körper nach Objekten mit der angegebenen Materialeigenschaft. Hat es welche gefunden, muß schließlich der Name des neuen Körpers genannt werden.

»Def_Mat« wird meistens erst dann gebraucht, wenn schon eine komplexe Szene mit vielen Körpern und Objekten geschaffen wurde. Sie haben vielleicht schon ein paar Testbilder »geraytraced« und sind zum Beispiel der Meinung, daß das Material »Blau-Metall« nicht in die Szene paßt. Mit »Def_Mat« können Sie leicht alle Objekte, die das Material »Blau-Metall« haben, heraussuchen und ihnen ein anderes Material zuordnen.

Körpername »*«
Bei vielen Funktionen, die einen neuen Körper bilden, fragt »Construct« nach einem Namen. Wenn Sie es sich anders überlegen, können Sie als Namen einfach ein »*« eingeben. In dem Fall wird kein neuer Körper erzeugt. Falls aber bereits Objekte geschaffen wurden (im »Tools«-Menü), dann hängen diese jetzt »in der Luft«. Lediglich über den »all«-Körper können Sie noch auf sie zugreifen.

Am Ende dieses Abschnitts noch ein Wort zu den Körpernamen. Geben Sie sich die Mühe, geeignete Namen auszudenken. Das klingt zwar banal, aber ich weiß zu meinem Bedauern, wie sehr man dazu neigt, beim Eintippen aus Faulheit schnell »irgendetwas« einzugeben. So wimmelt es dann nach kurzer Zeit im Auswahlfenster von Körpern mit Namen wie »xx«, »abc«, »123«, »i« und ähnlichem, und Sie wissen dann nicht mehr, welcher Körper »xx« ist. Das gleiche Problem hat man übrigens als Programmierer beim Ausdenken von Variablennamen. Bei »Construct« müssen Sie den Namen nur einmal eintippen; danach brauchen Sie den Körper nur anzuklicken; Namen wie »Sofa«, »Sitzgruppe«, »Linkes_Bein«, »Tragfläche«, »Baum« sind doch wirklich aussagekräftiger als »sf«, »sg«, »lb«, »trgfl« oder »bm«. Also, machen Sie sich die Mühe und benutzen Sie den Namen zur Beschreibung des Körpers.

3.5 Geometrische Veränderungen an Körpern

In diesem Abschnitt werden alle Funktionen aus dem Menü »Geomet« besprochen. In ihm sind alle Möglichkeiten zur geometrischen Manipulation von Körpern zusammengefaßt.

Körper sind, wie Sie inzwischen wissen, Mengen von Grundobjekten, also Dreiecken und Kugeln. Geometrisch verändert werden nicht die Körper selbst, sondern die Objekte, aus denen Sie bestehen. Objekte sind Dreiecke und Kugeln. Was aber wird an den Dreiecken verändert? Genaugenommen nichts, denn ein Dreieck ist nur eine »Sammlung von drei Eckpunkten«. Intern enthält das Dreieck drei Zeiger auf je einen Eckpunkt. Verändert werden nur die Eckpunkte.

Bei Kugeln ist der Sachverhalt etwas anders. Eine Kugel enthält einen Zeiger auf einen Mittelpunkt sowie den Radius. Verändert wird in der Regel der Mittelpunkt, bei Vergrößerungen aber auch der Radius.

Wenn man also von den Kugeln absieht, werden letztlich alle Funktionen nur an Punkten vorgenommen. Die Körper und Objekte dienen nur zur Bestimmung der betroffenen Punkte.

Die Funktionen sind alle Körper-orientiert, das heißt, Sie werden in der Regel keine einzelnen Objekte, sondern immer ganze Mengen von Objekten manipulieren.

Aber nun zu den einzelnen Funktionen, die in diesem Menüpunkt gesammelt sind:

Verändern – Verschieben, Vergrößern, Verzerren von Objekten.

Drehen – Rotieren von Körpern

Knittern – Fractalisieren/Zerknittern von Körpern

Die Funktionen sind alle grafisch orientiert, Sie führen sie mit der Maus am Plotfenster aus. Dazu müssen Sie allerdings mit den Darstellungsarten vertraut sein (Abschnitt 3.3).

Mit den geometrischen Veränderungen haben Sie sehr viele Möglichkeiten, Körper zu variieren, aber auch zu »zerstören«. Dies liegt daran, daß Sie absolute Freiheit bei der Definition von Körpern haben. Verändern Sie hier einen Körper, der Teil eines anderen Körpers ist, werden Sie den anderen zumindest arg verzerren. Sie müssen also selbst aufpassen, mit welchem Körper Sie welche Operation ausführen; »Construct« kann nicht wissen, ob eine Verzerrung, die Sie einem Körper zufügen, beabsichtigt oder ein Versehen ist. Nur bei Körpern, die Textur enthalten, muß »Construct« streng darauf achten, daß diese nicht zerstört oder verzerrt werden, und deswegen werden teilweise Operationen, die diese Körper betreffen, verboten. (Siehe auch 3.8.)

Verändern

Verändern ist ein recht allgemeiner Begriff, ich habe ihn aber gewählt, weil diese Funktion eigentlich aus drei Teilfunktionen besteht: Verschieben, Vergrößern/Verkleinern, Verzerren.

Nun kommt wieder das Rechteck ins Spiel, das bisher schon mehrfach verwendet wurde. Sie erkennen wohl schon, wieso ich die drei Funktionen in eine zusammengepackt habe. Die Handhabung des Rechteckes beinhaltet bereits Funktionen zum Verschieben/Vergrößern/Verzerren. Der Unterschied zwischen Vergrößern und Verzerren ist folgender:

Beim Vergrößern wird ein Körper in allen drei Koordinatenachsen gleichermaßen vergrößert. Sie erkennen das am Rechteck, das entsprechend der Mausbewegung seine Größe, aber nicht die Form verändert.

Beim Verzerren können Sie das Rechteck auf dem Plotfenster in die Breite und Höhe unabhängig verändern. Dies entspricht der Veränderung, die am Körper vorgenommen wird. Die Breite und Höhe des Körpers wird entsprechend der Breite und Höhe des Rechtecks verändert, während die Tiefe unverändert bleibt. In welche Richtung ein Körper verzerrt wird, hängt von der Blickrichtung ab. Bei Darstellung von vorn wird der Körper in Ost-West- und Hoch-Runter-Richtung verzerrt, während die Nord-Süd-Richtung nicht berührt wird. Bei Darstellung von links werden Nord-Süd- und Hoch-Runter verzerrt, wohingegen Ost-West unverändert bleiben. Probieren Sie das aus verschiedenen Richtungen aus, um ein Gespür für die Möglichkeiten zu bekommen.

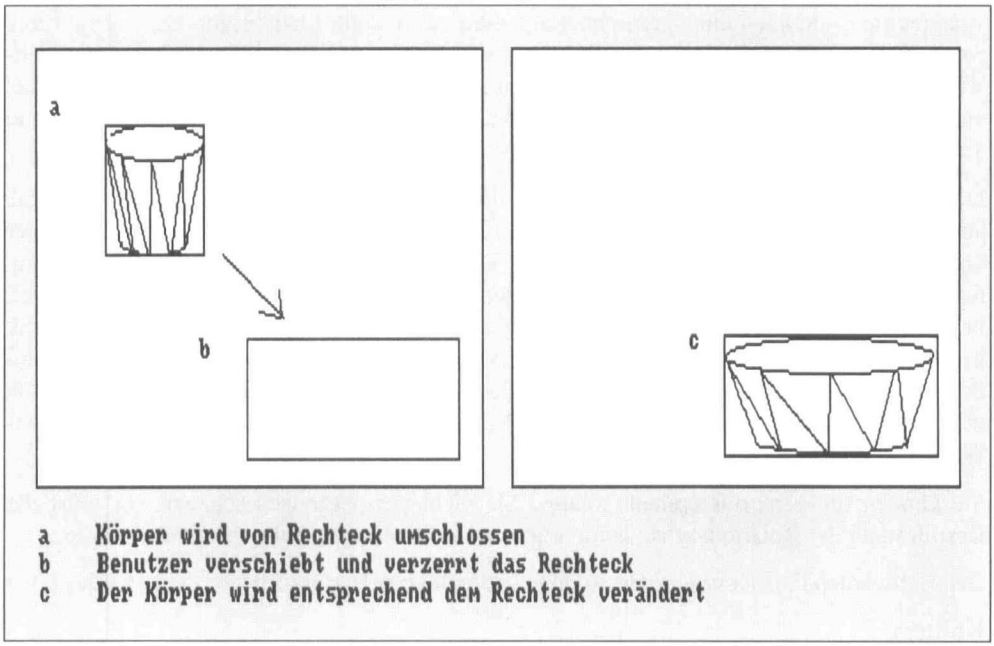


Bild 3.12: Verändern von Körpern

Nun zur Handhabung von »Verändern«: Bei Anwahl dieses Punktes fragt »Construct«, welchen Körper Sie verändern wollen. Nachdem Sie Ihre Wahl im Auswahlfenster getroffen haben, will »Construct« wissen, ob der Körper verzerrt oder vergrößert werden soll (bei Körpern, die Textur enthalten, funktioniert nur Vergrößern). Anschließend wird der Körper auf dem Plotfenster gezeichnet, falls er nicht schon vorher da war, und ein Rechteck wird so um den Körper gelegt, daß es ihn minimal umschließt. Sie können jetzt das Rechteck mit der Maus bewegen und vergrößern/verzerren, bis Sie mit Größe und Lage zufrieden sind. Danach führt »Construct« die Veränderung, die Sie am Rechteck vorgenommen haben, im Körper aus. Bild 3.12 verdeutlicht das Verändern von Körpern.

Drehen

»Drehen« ist eine Funktion, dessen Wirkung durch den Namen erschöpfend erklärt wird: Körper sollen gedreht werden. Wie bei »Verändern« muß zuerst der Körper angegeben werden, der gedreht werden soll. Der wird dann im Plotfenster gezeichnet. Jetzt hat man zwei Möglichkeiten: Entweder läßt man den Körper um seinen Mittelpunkt oder aber um einen beliebigen Punkt auf dem Plotfenster drehen. »Construct« fragt, was für eine Möglichkeit Sie vorziehen. Wollen Sie einen Drehpunkt angeben, müssen Sie mit dem Mauszeiger zu dem Punkt fahren und mit der linken Maustaste klicken, ansonsten nimmt »Construct« den Körpermittelpunkt als Drehpunkt. Vom Drehpunkt aus wird jetzt eine waagrechte Linie nach links gezeichnet und eine zweite zum Mauszeiger. Die zweite Linie folgt ständig dem Mauszeiger. Der Winkel zwischen beiden Linien entspricht dem Drehwinkel. Außerdem wird ständig der Drehwinkel angezeigt. Sobald Sie mit dem Drehwinkel zufrieden sind, klicken Sie mit der linken Maustaste. Die Drehung wird dann umgehend am Körper ausgeführt und der rotierte Körper wird neu gezeichnet (Bild 3.13).

Die Angabe des Drehpunktes ist in vielen Situationen praktisch, vor allem bei Körpern, die aus vielen Teilkörpern zusammengesetzt sind. Stellen Sie sich zum Beispiel einen Roboter vor, dessen Greifarm gedreht werden muß. Drehen Sie ihn um seinen Mittelpunkt, dann hängt er nicht mehr an seinem Gelenk, sondern schwebt frei in der »Luft«. In diesem Fall können Sie den Drehpunkt ins Gelenk setzen, und der Greifarm wird sauber um das Gelenk gedreht. Eine andere Frage ist, um welche Achsen gedreht wird. Wie schon bei »Verändern« ist dies durch die Blickrichtung festgelegt. Schauen Sie von vorne (von Süden), dann wird um die Nord-Süd-Achse gedreht, bei Blickrichtung von links (Westen) um die Ost-West-Achse usw.

Sie können übrigens nur drehen, solange Sie nicht perspektivisch schauen, da sonst die Bestimmung der Rotationsachse kompliziert ist und nicht mehr sehr anschaulich wäre.

Gedreht werden darf jeder Körper, solange dadurch kein Texturkörper verzerrt wird.

Knittern

Die Körper, die bisher erzeugt wurden, sind in der Regel schön rund, gleichmäßig oder erfüllen irgendwelche ästhetischen Symmetrieansprüche. Das muß nicht immer erwünscht sein. Viele Gegenstände in der Realität sind ganz und gar nicht symmetrisch oder glatt; von natürlichen Dingen, wie Pflanzen oder Felsen ganz zu schweigen.

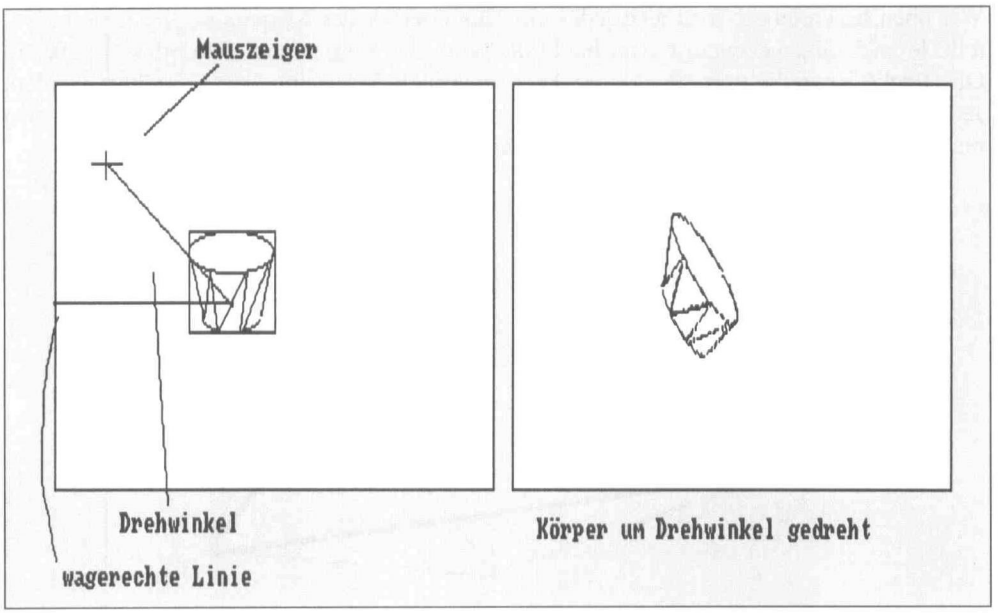


Bild 3.13: Rotieren eines Körpers

Mit Knittern haben Sie eine Operation an der Hand, mit der Sie Ihre bisher so glatten Körper buchstäblich zerknittern können. Allerdings werden nur die Dreiecke eines Körpers zerknittert, die Kugeln bleiben von dieser Operation unberührt.

Anhand eines Dreiecks soll der »Knittervorgang« verdeutlicht werden:

Man hat ein Dreieck mit den Eckpunkten 1, 2 und 3. Man erzeuge drei neue Punkte, indem jeweils die Mittelpunkte der Verbindungslinien 1–2, 2–3, und 3–1 um eine gewissen Wert verschoben werden.

Mit den neuen Punkten 4, 5 und 6 und den alten 1, 2 und 3 werden dann, wie in Bild 3.14 verdeutlicht, vier neue Dreiecke definiert. Die neuen Punkte sind durch Verschiebung der Kantenmittelpunkte des ursprünglichen Dreiecks entstanden. Der Grad der Verschiebung bestimmt das Aussehen des zerknitterten Körpers, der ja ein bißchen chaotisch werden soll.

Die Verschiebung jedes Kantenmittelpunktes wird folgendermaßen ermittelt:

Verschiebung = Zufall * Kantenlänge * Faktor * Richtung

»Zufall« ist eine Zufallszahl zwischen 0 und 1. »Kantenlänge« ist evident. »Faktor« wird von Ihnen eingegeben und bestimmt die Rauhigkeit des zerknitterten Körpers, und »Richtung« enthält die Verschiebungsrichtung.

Wie oben beschrieben, wird jetzt jedes einzelne Dreieck des Körpers in vier neue aufgeteilt. Je nach eingegebenem Faktor bleibt die Form des Körpers ungefähr erhalten, nur die Oberfläche ist zerknittert. Das ganze kann mehrmals hintereinander ausgeführt werden. Jedesmal werden alle Dreiecke des Körpers vervierfacht. Sie können den Vorgang also nicht zu oft wiederholen, da die Anzahl der Dreiecke dann gigantisch wächst.

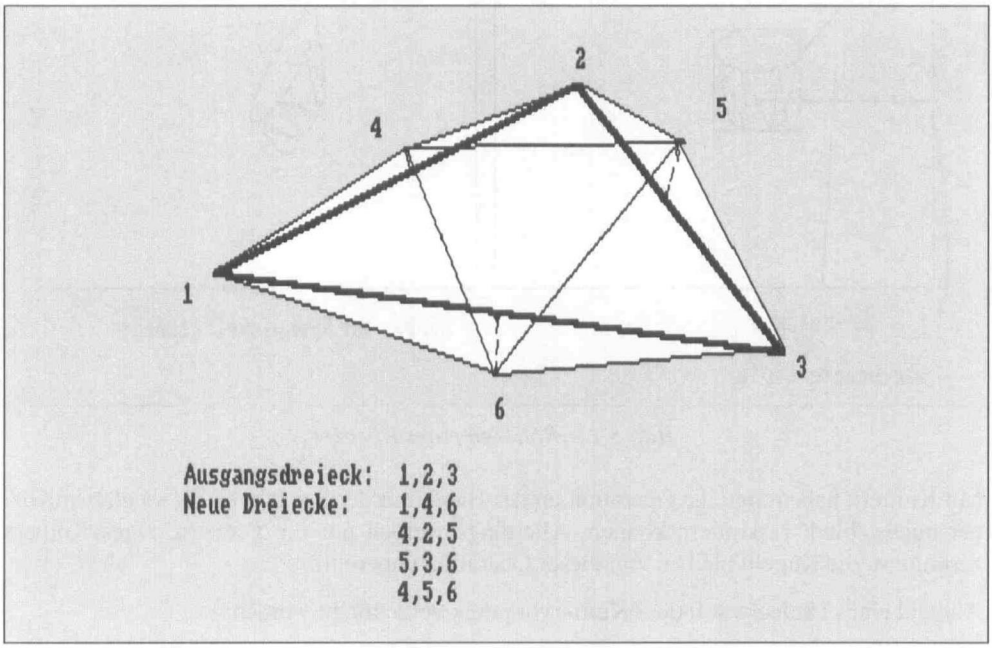


Bild 3.14: Zerknittern eines Dreiecks

Nun zur Handhabung von Knittern:

Bei »Anwahl« müssen Sie im Auswahlfenster den Körper anklicken, der zerknittert werden soll (Körper mit Textur werden abgelehnt). Danach müssen Sie sich zwischen zwei Modi entscheiden, die die Verschiebungsrichtung der Kantenmittelpunkte bestimmen. Bei »Modus1« werden die Punkte immer nach oben verschoben (Sie müssen in diesem Fall zuvor dafür gesorgt haben, daß der Körper in der richtigen Lage ist). Bei »Modus0« bestimmt »Construct« den Mittelpunkt des Körpers. Die Verschiebung der Punkte erfolgt dann in Richtung vom Mittelpunkt zum betreffenden Punkt.

Als nächstes muß der Faktor angegeben werden. Er bewegt sich zwischen »-0.3« und »0.3«. Bei Faktor »0« wird überhaupt nicht verschoben. Der Körper wird dann nur in viele kleine Dreiecke aufgeteilt, aber in der Form nicht verändert.

Danach erfolgt die Aufteilung der Dreiecke. Bei allzu umfangreichen Körpern wird die Aufteilung abgelehnt. Ansonsten wird der Körper, nachdem er zerknittert wurde, neu gezeichnet. Farbbild 2 zeigt links einen normalen Würfel und rechts einen, der zweimal zerknittert wurde.

3.6 Materialeigenschaften

In Abschnitt 3.4 wurde schon behandelt, wie Sie einem Körper eine Materialeigenschaft geben können. Nun soll genauer auf die Eigenschaften eingegangen werden; was sie bedeuten, wie man sie beeinflussen kann und schließlich, wie Sie neue Materialeigenschaften kreieren können. Mit den Materialeigenschaften kommen wir in Regionen, die schon etwas näher beim Raytracing angesiedelt sind. Bisher haben wir uns zum größten Teil nur um Dinge gekümmert, die mit der Geometrie der Szene zu tun haben. Jetzt werden wir den Objekten Eigenschaften verleihen, die sich beim Raytracing auswirken werden. Für diesen Abschnitt ist es übrigens nützlich, wenn Sie sich etwas mit dem Beleuchtungsmodell vertraut machen (Abschnitt 7.8).

Was für Eigenschaften kann man Objekten zuordnen? Nun, wir wollen farbige Bilder erzeugen, also muß zumindest Farbe im Spiel sein. Beim Raytracing faszinieren vor allem spiegelnde und durchsichtige Objekte, dergleichen muß also auch beachtet werden. Schließlich haben Gegenstände aus der realen Welt eine Oberflächenbeschaffenheit, sie sind rauh oder glatt, was sich beispielsweise beim Glanzlicht auswirkt.

Allerdings brauchen Sie sich nicht unbedingt mit dergleichen zu beschäftigen, um Objekten Materialeigenschaften zu geben. »Construct« hat eine Grundbibliothek von Eigenschaften, die Sie vergeben können und die für den Anfang sicherlich ausreicht. Die Materialnamen dürften schon ungefähr erklären, wie das Material später beim Raytracing erscheint. Wenn Sie sich aber für genauere Einzelheiten interessieren oder gar eigene Materialien schaffen wollen, sollten Sie sich mit den Funktionen im Menüpunkt »Material« beschäftigen.

Bevor wir zu den einzelnen Funktionen kommen, sollte zuerst mal aufgeführt werden, wie in Reflections eine Materialeigenschaft definiert ist:

- Ein Material besteht aus einer Grundfarbe (je drei Werte für Rot, Grün und Blau), einer Oberfläche und einer Textur. Zusätzlich hat das Material einen Namen, unter dem es dann zugreifbar ist.
- Eine Oberfläche hat fünf Faktoren für
  - *diffuse Reflektion*: Licht, das von den Lichtquellen kommt und in alle Richtungen verteilt wird.
  - *spiegelnde Reflektion*: Licht, das von anderen Objekten kommt und an der Oberfläche reflektiert wird.
  - *Lichtbrechung*: Licht, das die Oberfläche durchdringt und dabei gebrochen wird.

- *Eigenleuchten*: Licht, das ohne äußere Quellen von der Oberfläche ausgeht.
- *Brechungsindex*: Ein Maß, wie stark bei Lichtbrechung ein Strahl in der Richtung geändert wird.

Zusätzlich muß das Glanzlicht definiert werden, das einen starken Einfluß auf das Erscheinungsbild der Oberfläche in bezug auf die Rauheit hat.

- Eine Textur enthält ebenfalls einige Größen, die Sie beeinflussen können (werden in 3.7 erklärt), von denen die wichtigste die Texturdatei ist. Die Texturdatei ist eine IFF-Bilddatei, die das Texturmuster enthält.

Diese drei Bestandteile (Material, Oberfläche und Textur) können Sie getrennt editieren und sich daraus neue Materialeigenschaften zusammensetzen.

Beim Start kennt »Construct« nur ein Material, das »nichts« genannt wird und eine Oberfläche mit Namen »nichts_Oberfl«. Dieses Material ist immer definiert. Es hat die besondere Eigenschaft, daß beim Raytracing Objekte mit der Eigenschaft »nichts« nicht zu sehen sind. Die Frage, wozu so ein Material nützen soll, ist berechtigt. Falls Sie vergessen, irgendwelchen Objekten eine Materialeigenschaft zu geben, oder falls Objekte eine Eigenschaft haben, die in der Materialdatei nicht vorkommt, gibt »Beams« diesen Objekten automatisch die »nichts«-Eigenschaft. Zum anderen ist diese Eigenschaft im Zusammenhang mit Texturen gut zu gebrauchen, dann nämlich, wenn man Teile aus einem Objekt herausstanzen will (siehe 3.8 Material-Textur). Ansonsten sollte man nach dem Start von »Construct« mit »LoadMat« eine Materialdatei laden.

EditMat, NeuMat

Die beiden Funktionen sind prinzipiell die gleichen, nur wird bei »NeuMat« zuvor Platz für ein neues Material geschaffen, und das neue Material wird dann editiert.

Wie schon oben erläutert, besteht ein Material aus Farbe, Oberfläche und Textur. Von diesen drei Bestandteilen sind Farbe und Oberfläche unbedingt nötig, während die Textur weggelassen werden kann. Außerdem hat das Material einen Namen, unter dem Sie es ansprechen können.

Bei Anwahl von »EditMat/NeuMat« füllt sich das Plotfenster mit den folgenden Gadgets (siehe Bild 3.15).

Die linken vier Gadgets bedeuten:

- Im obersten Feld steht der Name des Materials. Bei neuen Materialien ist als Name »nichts« eingetragen. Wenn Sie das Feld anklicken, fragt »Construct« nach einem neuen Namen für das Material. Sie sollten hier allerdings einen eindeutigen Namen wählen. Erstens weiß man dann bei Vergabe ungefähr, was das Material für Eigenschaften hat, zweitens unterscheidet »Construct« die Materialien nach ihrem Namen. Das wirkt sich beim Dazuladen verschiedener Materialdateien aus (siehe Kap. 3.10).

- Im nächsten Feld (von oben) steht der Name der zum Material gehörenden Oberfläche. Auch hier steht »nichts_oberfl«, wenn Sie ein neues Material beginnen. Bei Anklicken des Feldes will »Construct« die neue Oberfläche wissen, die das Material bekommen soll. Sie können hier aber keinen neuen Namen erfinden, denn »Construct« will den Namen einer existierenden Oberfläche. Also erscheint das Auswahlfenster und Sie müssen die betreffende Oberfläche dort anklicken.

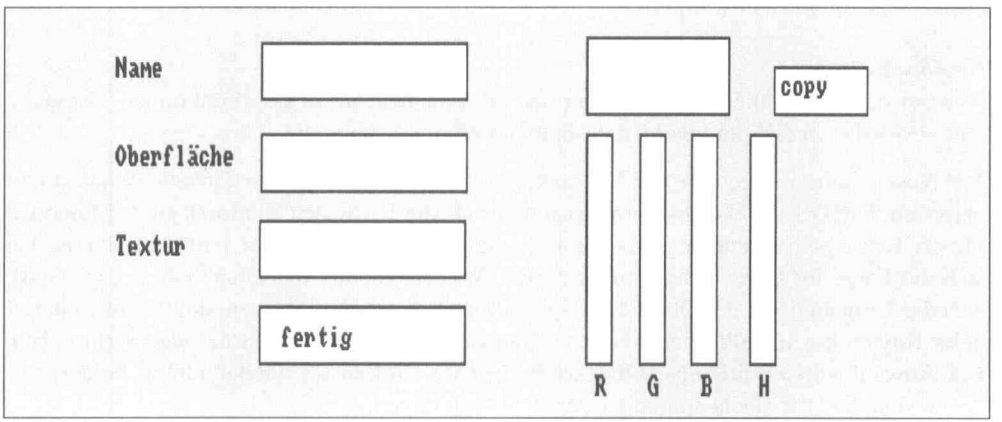


Bild 3.15: Die Gadgets nach Anwahl von »EditMat/NeuMat«

- Darunter steht ein Feld für die Textur des Materials. Die meisten Materialien haben keine, für die steht dort nur »keine Textur«. Um dem Material eine Textur zu geben, muß das Feld angeklickt werden, worauf wieder das Auswahlfenster, diesmal mit allen existierenden Texturen, erscheint, aus dem Sie durch Klick auswählen müssen. Dieses Feld darf bei einem existierenden Material aber nicht mehr geändert werden. »Construct« verhindert das, weil es sonst bei der Verwaltung der Texturen durcheinander käme.
- Mit dem untersten Feld links können Sie diese Funktion verlassen. »Construct« fragt, ob die Änderungen gesichert werden sollen.

Rechts finden Sie vier vertikale Balken. Es sind Schieberegler, mit denen Sie die Grundfarbe des Materials festsetzen können. Sie müssen nur den Mauszeiger in den jeweiligen Balken an die gewünschte Position bringen und kurz mit der linken Taste klicken. Der Regler bewegt sich dann in die entsprechende Position, und je nach Balken verändert sich die Rot-, Grün- oder Blaukomponente der Materialfarbe. Mit dem rechten Regler können Sie die Helligkeit der Farbe beeinflussen. Die eingestellte Farbe können Sie in dem Feld über den Schiebereglern sehen.

Rechts neben dem Farbfeld befindet sich ein »Copy«-Feld, mit dem Sie Materialien kopieren können.

Nachdem Sie das Material eingegeben und gesichert haben, können Sie es von nun an unter seinem Namen an Körper vergeben.

Matweg
Hiermit können Sie ein Material wieder löschen. Es ist aber nicht empfehlenswert, eines zu löschen, das schon an Objekte vergeben wurde. In diesem Fall bekommen die entsprechenden Objekte automatisch das Material »nichts«. Das erste Material, nämlich »nichts«, kann übrigens nicht gelöscht werden.

NeuOb, Editob
Wie bei den Materialien sind auch bei den Oberflächen die beiden Funktionen »NeuOb« und »EditOb« ähnlich und werden deshalb nur einmal behandelt.

Mit dieser Funktion werden alle Parameter beeinflußt, die die Oberflächenbeschaffenheit betreffen. Die Oberfläche bestimmt noch mehr als die Farbe den Eindruck eines Materials. Sie als Betrachter erkennen ein bestimmtes Material weniger an der Farbe als daran, wie sich die Farbe im Oberflächenverlauf ändert. Wie ausgeprägt das Glanzlicht ist, wie stark sich die Umgebung in der Oberfläche spiegelt – das sind die Faktoren, die einem Material beim Raytracing metallischen »Touch« oder die Stumpfheit von Holz oder Stein geben. Bei Anwahl von »EditOb/NeuOb« erscheinen im Plotfenster wieder einige Felder, mit denen sich die Oberflächenparameter beeinflussen lassen.

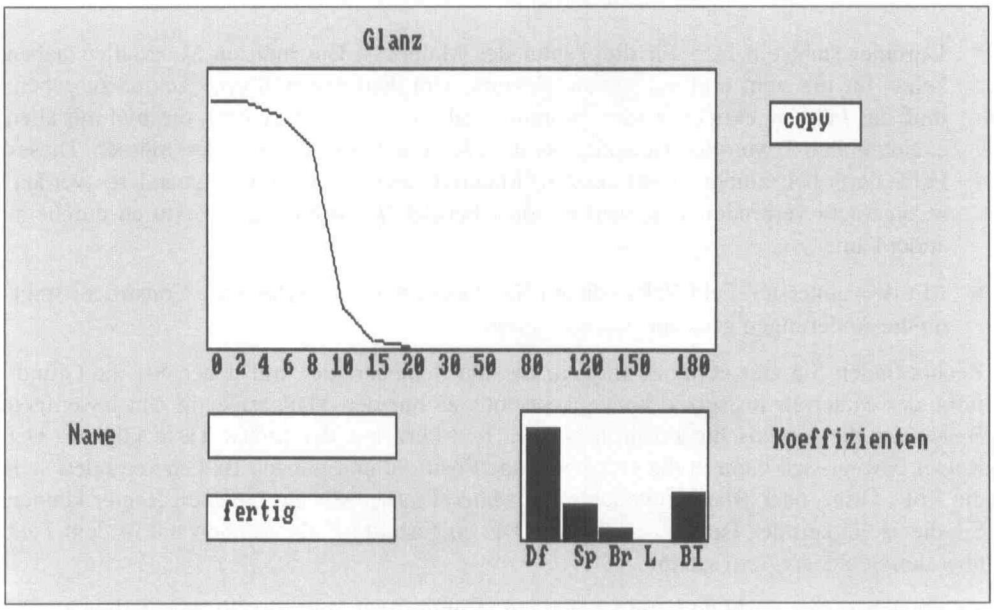


Bild 3.16: Die Gadgets nach Anwahl von »EditOb/NeuOb«

Das große Feld enthält den Verlauf der Stärke des Glanzlichtes in Abhängigkeit vom Winkel zwischen den Vektoren R und L (siehe 7.8). Schauen Sie sich einmal die Glanzkurven für verschiedene Oberflächen an. Es ist leicht zu erkennen, daß bei glatten Oberflächen wie Metall oder Glas die Glanzkurve sehr rasch steil auf Null abfällt. Bei weniger glatten Oberflächen hingegen fällt sie sanfter ab, wie Sie etwa bei »mattes-Blau« erkennen können.

Die Kurve kann sehr einfach verändert werden: Bewegen Sie den Mauszeiger in das Feld mit der Glanzkurve. Solange die linke Maustaste gedrückt bleibt, folgt die Glanzkurve der Bewegung des Mauszeigers.

Die beiden linken Felder darunter sind zur Eingabe des Oberflächennamens sowie zum Sichern der geänderten Parameter gedacht, genau wie bei »Material«. Für den Oberflächennamen gilt im übrigen das gleiche wie für den Materialnamen.

Mit dem Feld rechts unten werden die fünf Parameter »diff«, »spieg«, »brech«, »leucht« und »brech_index« beeinflußt.

Im Beleuchtungsmodell von Reflections (siehe auch 7.8) setzt sich die Farbe, die auf der Oberfläche eines Objektes zu sehen ist, aus vier Anteilen zusammen (eigentlich fünf, wenn man das Glanzlicht dazuzählt):

- *Diffuser Farbanteil (Df).* Dieser berechnet sich aus der Grundfarbe des Objekts und der Farbe der Lichtquelle(n), die unter Berücksichtigung von Oberflächennormale und Richtung zur Lichtquelle zusammengerechnet werden.
- *Spiegelnder Farbanteil (Sp).* Dies ist die Farbe, die man bekommt, wenn man einen Strahl von der Objektoberfläche in Richtung des Reflektionsvektors losschickt. (Also entweder die Hintergrundfarbe oder die Farbe eines anderen Objektes.) Dieser Anteil ist nur bei spiegelnden Oberflächen von Bedeutung.
- *Brechender Farbanteil (Br).* Entspricht dem spiegelnden Farbanteil, nur wird diesmal ein Strahl in die Brechungsrichtung abgeschickt. Dieser Anteil gilt nur für durchsichtige Objekte.
- *Eigenleuchten (L).* Der eigenleuchtende Farbanteil berechnet sich wieder aus der Grundfarbe des Objektes. Hierbei werden aber Lichtquellen nicht berücksichtigt. Dies simuliert ein leuchtendes Objekt, allerdings ist es keine echte Lichtquelle, die Schatten wirft.
- *Brechungsindex (BI).* Dieser Index bestimmt bei durchsichtigen Objekten, wie stark der Strahl beim Durchgang durch die Oberfläche gebrochen wird.

Die jeweilige Stärke der vier Anteile wird durch vier Faktoren festgelegt, mit der die Anteile - jeweils gewichtet - zusammengerechnet werden. Die Summe der vier Anteile muß zusammen 1 ergeben (aber dafür sorgt »Construct« automatisch). Bei einer steinernen Oberfläche sind beispielsweise die Faktoren »Sp«, »Br« und »L« gleich 0, weil sich da nichts spiegelt und ein Stein auch nicht durchsichtig ist. Bei einem reinen Spiegel hingegen

wird der diffuse Farbanteil nahe bei Null liegen, während der spiegelnde dominiert. Entsprechend ist bei Glas der brechende am stärksten ausgeprägt.

Die vier Anteile können Sie in dem breiten Feld unter der Glanzkurve verändern. Die vier Balken in dem Feld entsprechen von links nach rechts den vier Farbanteilen (»diff«, »spieg«, »brech« und »leucht«). Daneben ist nochmals ein einzelner Schieberegler für den Brechungsindex. Wenn Sie den Balken ganz nach oben schieben, entspricht das keiner Brechung, das heißt, ein Strahl wird bei Durchgang durch die Oberfläche nicht verändert. Ist der Balken unten, entspricht das einer starken Brechung des Strahls.

Rechts neben dem Glanzfeld befindet sich noch ein weiteres, das mit »copy« beschriftet ist. Damit können Sie eine andere Oberfläche kopieren. Dies ist nützlich, wenn Sie eine neue Oberfläche erzeugen wollen, die einer vorhandenen weitgehend ähnelt. Sie kopieren sich die vorhandene und verändern nur die Parameter, die bei der neuen Oberfläche anders sein sollen.

Farbbild 3 zeigt drei Kugeln mit verschiedenen Oberflächeneigenschaften (die Grundfarbe ist gleich). Die linke ist metallisch, das heißt, sie spiegelt ein wenig und hat ein scharf ausgeprägtes Glanzlicht. Die mittlere Kugel ist matt, sie spiegelt nicht, hat aber ein Glanzlicht, das über einen größeren Bereich sanft abfällt. Die rechte Kugel ist ebenfalls matt, hat aber kein Glanzlicht.

Obweg

»Obweg« dient der Löschung einer Oberfläche. Es darf allerdings keine Oberfläche gelöscht werden, die noch in einem Material vorkommt. In dem Fall verhindert »Construct« das Löschen.

3.7 Texturen

Endlich kommen wir zu dem Thema, das bisher immer als »Störenfried« im Hintergrund wirkte, das ich aber bisher nicht näher erläutern wollte. Das hatte seine Gründe: Solange Sie keine Texturen verwenden, stören Sie auch nicht die übrigen Funktionen von »Construct«, sind also zum Verständnis nicht notwendig. Sie sollten sich zuerst mit »Construct« und seinen Funktionen vertraut machen, bevor Sie sich den Texturen zuwenden.

Vielleicht fragen Sie, wozu Texturen überhaupt notwendig sind, wenn Sie doch nur stören. Gewiß, notwendig sind Sie nicht unbedingt, aber die Möglichkeiten, die sich Ihnen mit Texturen bieten, werden Sie gewiß – genauso wie mich – derart faszinieren, daß Sie die kleinen Beeinträchtigungen in Kauf nehmen werden. Aber jetzt erst einmal zu grundlegenden Erläuterungen:

Eine Textur ist schlicht und einfach ein Muster. Muster kann man noch ganz allgemein auslegen, meist wird jedoch ein Farbmuster zugrundegelegt. Man hat ein Bild und nimmt die Farbe des Bildes als Grundfarbe eines Objektes. Damit bekommt das Objekt nahezu an jeder Stelle eine andere Farbe. Es ist, so glaube ich, leicht einsichtig, daß man schon damit einer Szene sehr viel mehr Realismus geben kann. Denken Sie sich zum Beispiel das Bild einer Mauerwand, das als Grundfarbe einem mit »Construct« erstellten Haus das Erscheinungsbild eines echten Gemäuers verleiht; bisher sah es ja lediglich wie ein detailarmer Klotz aus. Sie könnten das Haus auch aus vielen kleinen Ziegeln zusammenbauen, und nach viel Schweiß und Mühe erhielten Sie dann ebenfalls ein realistisch aussehendes Haus. Aber diese Methode würde einen viel größeren Aufwand beim Designen und eine gewaltige Anzahl von Objekten benötigen, was man vermeiden sollte, solange es nicht nötig ist. Die beiden Hauptvorteile von Texturen sind also folgende:

- Höherer Realismus des Bildes durch mehr Details, die als Oberflächenmuster auf den Objekten zu sehen sind.
- Einsparung von Objekten, da man schon durch Texturen viel simulieren kann, was sonst nur durch viele Objekte zustande käme.

»Construct« ist körperorientiert. Sie haben also normalerweise nicht mit einzelnen Dreiecken oder Kugeln zu tun, sondern mit Körpern, die ihrerseits Mengen von Dreiecken und Kugeln sind. Die Texturen schließen sich nahtlos an. Sie können also Texturen nur an ganze Körper vergeben. Angenommen, Sie hätten einen Körper aus einigen hundert Objekten und gäben diesem eine Textur. Wenn der Raytracer später auf eines der Objekte des Körpers stieße und aus dem Schnittpunkt die entsprechenden Pixel des Texturbildes ausrechnen wollte, wie sollte das dann funktionieren? Die Textur soll ja dem Körper als ganzem gehören und nicht jedem Objekt des Körpers für sich. Man muß also die Schnittpunktkoordinaten des Strahls mit dem Objekt in Bezug zu den Koordinaten des Körpers als ganzen setzen. Der Raytracer weiß allerdings nichts von Körpern; er kennt nur Dreiecke und Kugeln – bisher jedenfalls, denn ab jetzt kommen auch Textur-Objekte hinzu.

Sobald Sie in »Construct« einem Körper eine Textur geben, wird automatisch eine neue Datenstruktur angelegt. Die enthält Angaben über:

- Koordinaten, Größe und Orientierung des Körpers,
- das zugehörige Textur-Bild,
- einige Parameter über die Art der Textur-Abbildung (s.u.).

Die Datenstruktur nenne ich »Textur-Objekt«. Diese Textur-Objekte bekommt der Raytracer auch, und daraus kann er die nötigen Berechnungen vornehmen.

Das Besondere an den Textur-Objekten ist, daß sie in »Construct« alle geometrischen Veränderungen mitmachen, die an dem zugehörigen Körper vorgenommen werden. In »Construct« sind die Textur-Objekte ganz eng mit ihren Körpern verbunden. Das Errechnen der Pixelfarbe mit Textur-Objekten ist in Bild 3.17 gezeigt.

»Construct« prüft vor jeder Operation, ob durch sie eine Textur in irgendeiner Weise berühren könnte. Dabei werden grundsätzlich alle Arten von Verzerrungen verboten. Andere Operationen, wie Verschieben, Vergrößern, Drehen und Löschen, werden nur dann erlaubt, wenn der Texturkörper als Ganzer verändert wird und nicht Teile davon. Beispiel:

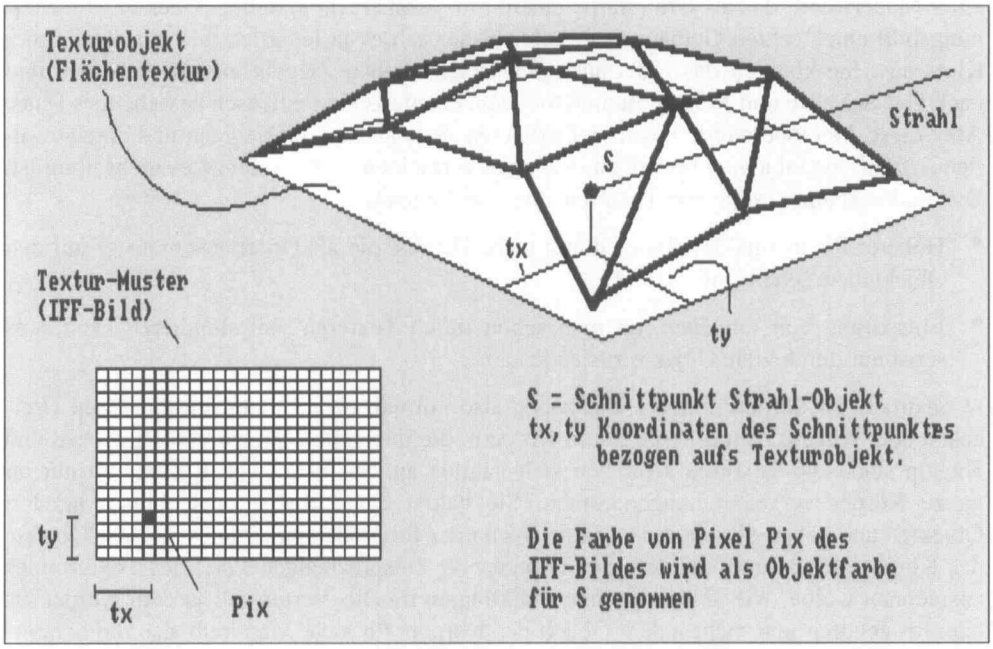


Bild 3.17: Errechnen der Pixelfarbe mit Textur-Objekten

Nehmen wir wieder unseren Körper »Sitzgruppe«, dem wir zum Beispiel ein Texturmaterial »Leder« vergeben haben. »Sitzgruppe« als Ganze darf noch verschoben, vergrößert und gedreht werden, auch ein Körper »Zimmer«, der »Sitzgruppe« vollständig enthält. Versuchen Sie aber den Körper »Sofa« oder »Sessel« zu verändern, so weigert sich »Construct« hartnäckig. Diese Restriktion liegt daran, daß es zu aufwendig wäre, jede Art von Verzerrung, die an Objekten vorgenommen werden kann, mit den Textur-Objekten nachzuvollziehen. Und das wäre nötig, damit der Raytracer eine korrekte Texturabbildung berechnen kann.

Texturarten

Die Texturabbildung kann auf sehr vielfältige Weise vorgenommen werden. In Reflections sind drei Abbildungsarten implementiert:

1) Flächentextur
Die bekannteste Texturart ist die Flächentextur. Bei ihr wird das Texturmuster wie bei einer Dia-Projektion auf dem Körper abgebildet. Dabei wird so getan, als ob der Körper als Ganzer ein Viereck wäre, was er aber nicht zu sein braucht. Sie können durchaus eine Flächentextur an einen Rotationskörper vergeben. Intern erzeugt »Construct« ein Viereck, das von jetzt an alle Bewegungen des Körpers mitmacht und später beim Raytracing eine korrekte Texturabbildung gewährleistet. Farbbild 4 enthält ein Texturmuster, das die verschiedenen Texturarten einmal demonstrieren soll. In Farbbild 5 ist das Testmuster auf einen Körper als Flächentextur projeziert worden. Daneben ist der gleiche Körper ohne Textur.

2) Kugeltextur
Die nächste Texturart, die in Reflections angeboten wird, ist die Kugeltextur. Die Abbildung können Sie sich hierbei am besten wie die Abbildung einer Landkarte auf einen Globus vorstellen. Intern wird jetzt eine Kugel erzeugt, die alle Bewegungen des Körpers nachvollzieht. Farbbild 6 zeigt, wie die Testtextur um eine Kugel gewickelt wurde.

3) Rotationskörpertextur
Eine Variante der Kugeltextur ist die Rotationskörpertextur. Dabei wird das Texturmuster um einen Zylinder gewickelt. Ein einprägsames Modell hierfür wäre eine Litfaßsäule, um die ein Texturmuster wie ein Plakat aufgeklebt wird. In Farbbild 7 wurde diese Texturart angewendet.

Man erzielt das beste Ergebnis, wenn man die jeweiligen Texturarten auch bei Körpern verwendet, die ungefähr die entsprechende Form haben; aber Sie haben freie Hand. Probieren Sie ruhig einmal aus, wie eine Kugeltextur bei einem Rotationskörper oder einem Quader wirkt oder wie sich eine Flächentextur auf einer Kugel ausmacht usw.

Neben der Texturart kann die Skalierung der Abbildung bestimmt werden, das heißt, wie groß das Texturbild auf dem Objekt erscheinen soll und wo es beginnen soll. Zu Beginn ist die Skalierung so eingestellt, daß das Texturbild den Körper vollständig bedeckt, aber Sie können die Skalierung so ändern, daß das Texturbild sehr viel kleiner oder größer ist, und außerdem kann der Ursprungspunkt des Bildes beliebig verschoben werden.

Vergabe von Texturmaterialien

Wie wird ein Texturmaterial einem Körper zugeteilt? Anfangs genauso wie jedes andere Material. Sie haben im »Körper«-Menü die Funktion »Material« angewählt und dort im Auswahlfenster für Materialien ein Texturmaterial angeklickt (die sind rot geschrieben im Gegensatz zu normalen Materialien, die in schwarzer Schrift dastehen). Ab jetzt ändert sich der Ablauf etwas bzw. es geht noch weiter, wenn man bei normalen Materialien bereits fertig ist. Zuerst will »Construct« nämlich wissen, welche der drei Arten (Flächen-,

Kugel- oder Rotationskörpertextur) man dem Körper geben will. Bei Flächentextur geht es folgendermaßen weiter:

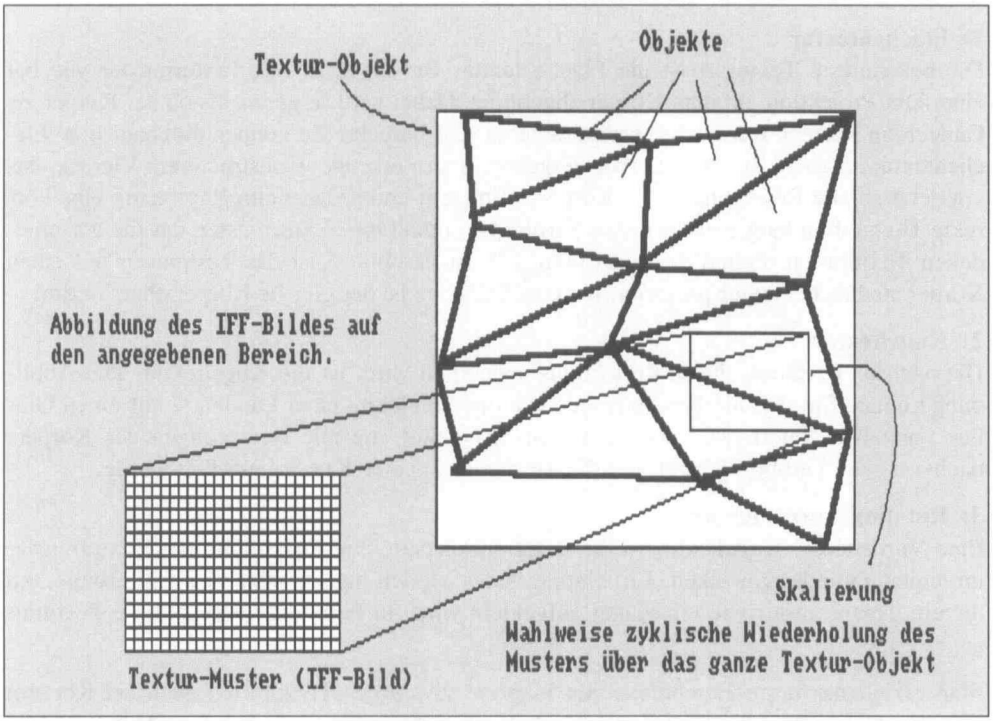


Bild 3.18: Skalierung bei Flächentextur

Der Körper wird neu gezeichnet und ein rotes Rechteck erscheint, das den Körper umschließt. Das Rechteck zeigt die Lage des Texturbildes auf dem Körper an. So, wie Sie jetzt das Rechteck sehen, wird später das Texturmuster daraufprojeziert. Wie schon bei anderer Gelegenheit können Sie jetzt das Rechteck in Höhe, Breite und Lage verändern. Die Projektionsrichtung wird durch Ihre Blickrichtung bestimmt. Schauen Sie von vorne, wird das Texturbild von Süden nach Norden auf den Körper projeziert, bei Blickrichtung von links von Westen nach Osten usw. Schauen Sie perspektivisch, wird zuvor automatisch der Blick von oben eingerichtet. Bild 3.18 entspricht der Skalierung bei Flächentextur.

Bei Kugel- und Rotationskörpertextur geht es prinzipiell genauso vor sich, nur wird die Definition der Skalierung grafisch etwas anders gehandhabt. Das liegt daran, daß die Abbildungsart hierbei (das »Umwickeln« des Körpers) interaktiv nicht so gut darstellbar ist wie bei der Flächentextur. Der Körper wird im linken Teil des Plot-Fensters neu gezeichnet. Die Rotationsachse bzw. die Nordpol-Südpol-Achse bei Kugeltextur geht durch die

Mitte des Körpers von unten nach oben (im Plotfenster). Rechts neben dem Körper erscheint das rote Rechteck und darüber eine waagerechte Skala von 0 bis 360 Grad. In seiner y-Ausdehnung entspricht das Rechteck genau der y-Ausdehnung des Körpers, und in seiner x-Ausdehnung genau 360 Grad. Das bedeutet, daß zu Anfang das Texturbild vollständig um den Körper gewickelt wird (360 Grad) und ihn entlang der Rotationsachse bzw. von Nordpol zu Südpol vollständig bedeckt. 0 Grad bzw. 360 Grad liegen genau in der Mitte des Körpers. 90 Grad bzw. 270 Grad entsprechen im Bild dem rechten bzw. linken Rand des Körpers. Bild 3.19 zeigt die Skalierung bei »Kugel/Rotk.«-Textur.

Wichtig ist bei allen drei Arten, daß Sie sich vorher überlegen, wie das Texturmuster auf dem Körper liegen soll, und dementsprechend die Blickrichtung wählen, aus der Sie den Körper plotten, denn dadurch wird bei Flächentextur die x- bzw. y-Achse und bei »Rotk.« die Rotationsachse des Texturobjektes festgelegt. Sieht nachher beim Raytracing die Textur nicht so aus wie beabsichtigt, könnte es daran liegen, daß Sie bei Texturvergabe den Körper aus einer falschen Richtung betrachtet haben.

Falls Sie das Rechteck verkleinern, wird das Texturbild automatisch auf dem Körper zyklisch wiederholt, und zwar sowohl in x- als auch in y-Richtung. Dies ist durchaus oft von Nutzen und beabsichtigt. Falls Sie aber wirklich nur den vom Rechteck bedeckten Teil des Körpers durch die Textur belegt haben wollen, müssen Sie eine »Etikett-Textur« (siehe unten) wählen.

Textureigenschaften

Außer durch die Abbildungsart und die Skalierung können Texturen aber noch durch weitere Parameter beeinflußt werden. Reflections bietet vier Eigenschaften an, die bestimmen, wie das Texturmuster von »Beams« interpretiert wird. Die Eigenschaften sind im einzelnen:

1) »Etik«
Etikett-Textur. Wenn Sie die Skalierung so verkleinern, daß nicht mehr der ganze Körper bedeckt wird, wird normalerweise das Texturmuster immer zyklisch wiederholt. Bei Etikett-Textur geschieht das nicht. Da bedeckt das Bild effektiv nur den Bereich, den Sie mit dem Rechteck anwählen. Der Rest bekommt die Grundeigenschaft des gewählten Materials. Das Etikett hingegen bekommt die erste Materialeigenschaft, die in der Materialtabelle (siehe unten) der Textur steht. Sie müssen also dafür sorgen, daß bei Etikett-Textur zumindest der erste Eintrag der Materialtabelle dieser Textur belegt ist.

2) »Genl«
Genlock-Textur. Der Name ist deshalb gewählt, weil in gewisser Hinsicht das gleiche Grundprinzip wie bei Genlock-Video besteht. Dort wird bei einem Bild für alle Pixel mit Farbe 0 nicht deren Farbe, sondern die einer externen Bildquelle (zum Beispiel ein Videosignal) genommen. Farbe 0 des Texturmusters läßt das Grundmaterial durchscheinen. Alle anderen Farben werden entsprechend der dem ersten Eintrag in der Materialtabelle der Textur behandelt. Wenn nur Teile des Körpers Textur erhalten sollen, ist Genlock-Textur

(neben Etikett-Textur) eine Möglichkeit dafür. Sie müssen nur Ihr IFF-Bild (das Textur-Muster) so setzen, daß die Bereiche, die später nicht zu sehen sein sollen, die Farbe 0 bekommen.

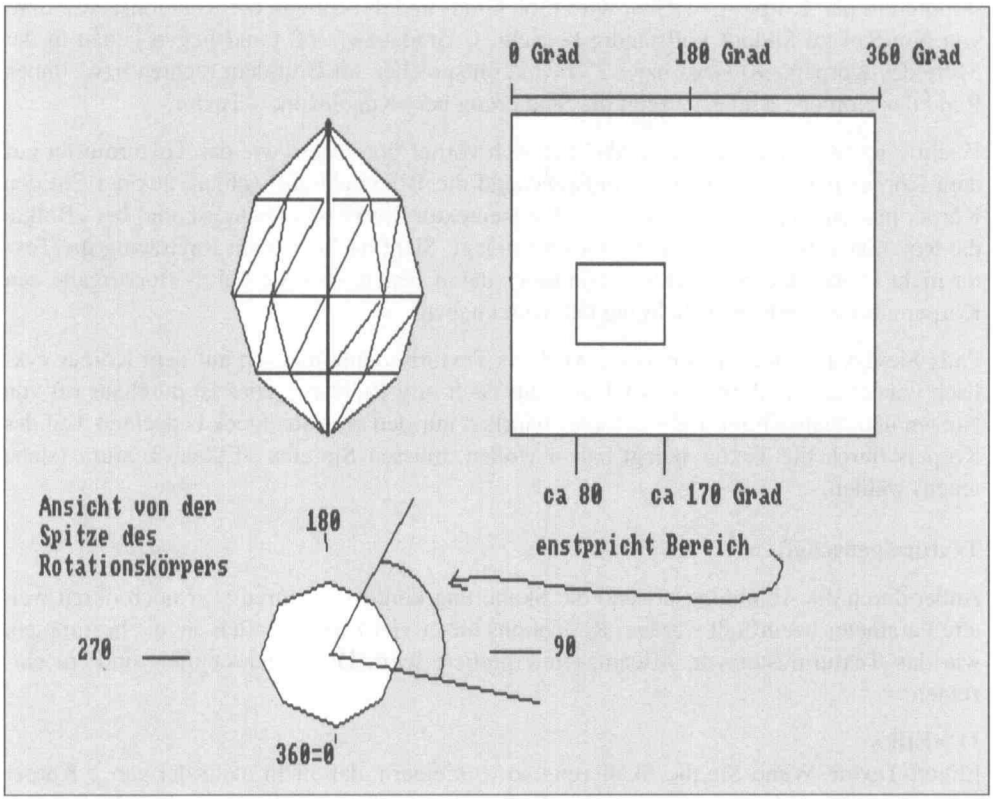


Bild 3.19: Skalierung bei »Kugel/Rotk.«-Textur

3) »Mat«

Material-Textur. Anstelle eines Farbmusters definiert das IFF-Bild ein Materialmuster. Die Pixelwerte werden nicht als Indizes in eine Farbtabelle, sondern als Indizes der Textur-eigenen Materialtabelle (siehe unten) gewertet. Sie können hiermit auf einem Körper beliebige Materialien als Muster kombinieren. Sie müssen bei der Definition der Textur (»EditTex/NeuTex«) die Materialtabelle der Textur mit den gewünschten Materialien füllen. Bild 3.20 zeigt den Unterschied zwischen Farb- und Materialtextur.

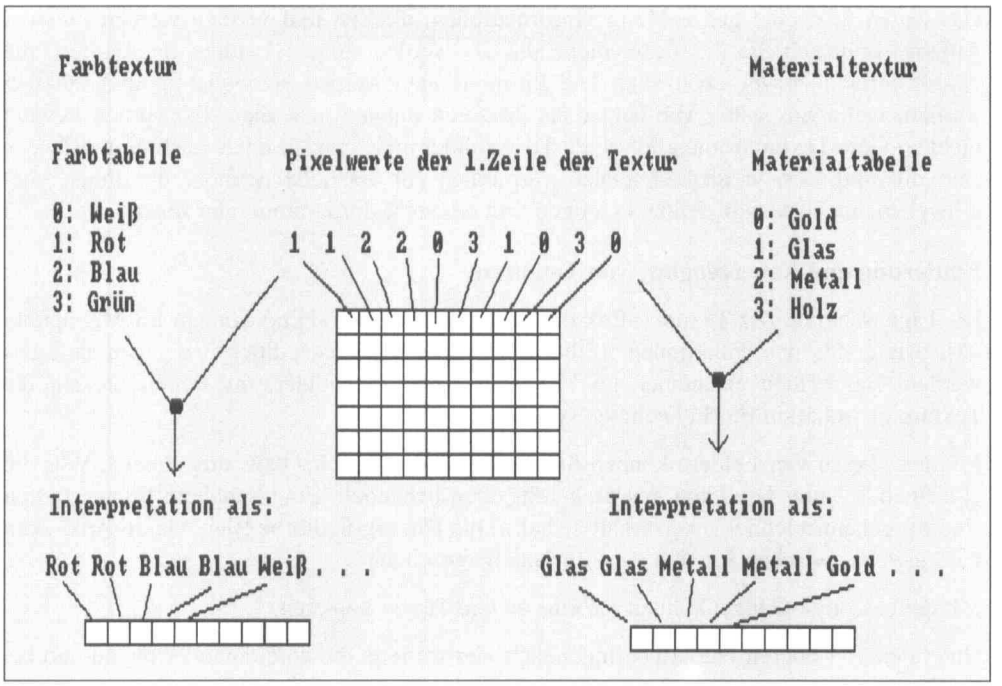


Bild 3.20: Der Unterschied zwischen Farb- und Materialtextur

4) »Pkt«

Punkt-Textur. Die Pixel des IFF-Bildes werden nicht als kleine Rechtecke behandelt, sondern als Punkte. Da die Punkte nie exakt getroffen werden, werden die Werte (Farbe oder Material) benachbarter Pixel (Punkte) interpoliert. Sie können damit innerhalb der Textur weiche Übergänge schaffen. Das ist besonders interessant, wenn mit der Kamera so nahe an den Körper gezoomt wird, daß beim Raytracing die einzelnen Pixel des IFF-Bildes erkennbar sind. Wenn das stört, kann dem mit Punkt-Textur abgeholfen werden. Ein weiterer Aspekt der Punkt-Textur ist die Möglichkeit, zwischen zwei Farben oder Materialien weiche Übergänge zu erzeugen. Hierzu empfehlen sich Texturen mit nur wenigen Pixeln.

Diese vier Eigenschaften können beliebig miteinander kombiniert werden. Zusätzlich kann eine Textur eine eigene Materialtabelle haben. Sie wird auf jeden Fall bei Material-Textur und auch bei Genlock- und Etikett-Textur verwendet. In ihr stehen bis zu 32 Verweise auf die aktuelle Materialtabelle. Ein IFF-Bild enthält für jedes Pixel einen Index für seine Farbtabelle. Bei Materialtextur wird der Index für die Materialtabelle anstatt für die Farbtabelle genommen. Beim Raytracing wird dadurch nicht nur die Farbe an der aktuellen Stelle geändert, sondern die gesamte Materialeigenschaft des Objektes. Beispiele dazu folgen in Kapitel 3.8.

Das waren jetzt eine ganze Menge Informationen, die erst mal verdaut werden müssen. Anfangs brauchen Sie die sicher nicht alle. Es werden einige Texturen mitgeliefert, die relativ einfach zu vergeben sind. Die Informationen werden aber wichtig und nützlich werden, wenn Sie selber IFF-Bilder als Texturen entwerfen wollen. Es wurden bewußt nicht so viele Texturen mitgeliefert, da Texturbilder meist nur für einen bestimmten Zweck sinnvoll sind. Sie werden sich also vermutlich für spezielle Szenen, die Ihnen vorschweben, auch spezielle Bilder erzeugen (mit einem Malprogramm oder sonstwie).

Editierung und Neuerzeugung von Texturen

Die Eigenschaften der Textur (»Etik«, »Genl«, »Mat« und »Pkt«) können im Menüpunkt »Material« mit den Funktionen »EditTex« bzw. »NeuTex« editiert bzw. neu definiert werden. Bei beiden erscheinen im Plotfenster folgende Felder, mit denen Sie auf die Textureigenschaften Einfluß nehmen können.

Mit den oberen vier Feldern können Sie die Eigenschaften ein- bzw. ausschalten. Was die einzelnen Schalter bewirken, wurde bereits oben behandelt. Eingeschaltete Eigenschaften sind rot gekennzeichnet, ausgeschaltete haben die Hintergrundfarbe Blau. Durch Anklicken wird jeweils zwischen Ein und Aus hin- und hergeschaltet.

Mit dem »Copy«-Gadget können Sie eine andere Textur kopieren.

Unter den vier oberen Feldern befinden sich vier weitere, die untereinander liegen und bei Anklicken folgendes bewirken:

Das oberste ist das »Mat_Tab«-Feld. Bei Anklicken erscheint im Listfenster die Materialtabelle dieser Textur. Danach wird gefragt, ob die Materialtabelle geändert werden soll. Wenn ja, können Sie nacheinander im Auswahlfenster die einzelnen Einträge der Materialtabelle anklicken. Sie müssen beachten, wie viele Einträge Sie in die Tabelle einbringen wollen, was wiederum von der Farbanzahl des IFF-Bildes abhängt. Alle Einträge, die Sie nicht setzen, werden mit dem Material »nichts« belegt. Fall Sie nicht wissen, wie viele Material-Einträge ein Bild haben darf, können Sie die feststellen, indem Sie das unterste Feld (»IFF-File«) anklicken. Darauf will »Construct« einen neuen Datei-Namen wissen, drücken Sie aber nur [Return], da der Name nicht geändert werden soll. »Construct« liest dann aus der Bild-Datei einige Parameter (s.u.), unter anderem die Anzahl der Farben des Bildes. Ebenso viele Material-Einträge dürfen Sie in die Materialtabelle setzen. Mit dem Feld darunter können Sie die Funktion abbrechen. »Construct« fragt noch, ob die Änderungen gesichert werden sollen.

Im nächsten Feld wird der Name der Textur bestimmt. Hier können Sie durch Anklicken einen neuen Texturnamen vergeben. Dieser Texturname hat nichts mit dem Namen der dazugehörigen IFF-Datei zu tun. Sie können ihn beliebig wählen. Es empfiehlt sich aber, bei der Namensgebung Bezug auf die Eigenschaften zu nehmen, die Sie dieser Textur geben, so daß Sie bei der Texturvergabe durch den Namen ungefähr über die Eigenschaften der Textur orientiert sind.

Ganz unten müssen Sie den Namen des dazugehörigen IFF-Bildes eingeben. Es darf nur der einfache Dateiname sein. Die IFF-Datei muß im Ordner »Reflections/text/« stehen. Dort sucht »Beams« später das Bild. Ist es dort nicht gespeichert, bricht »Beams« ab. Wenn Sie einen neuen Datei-Namen angeben, prüft »Construct«, ob die Datei im Ordner »Reflections/text/« existiert. Wenn ja (und falls es eine korrekte IFF-Bilddatei ist), liest

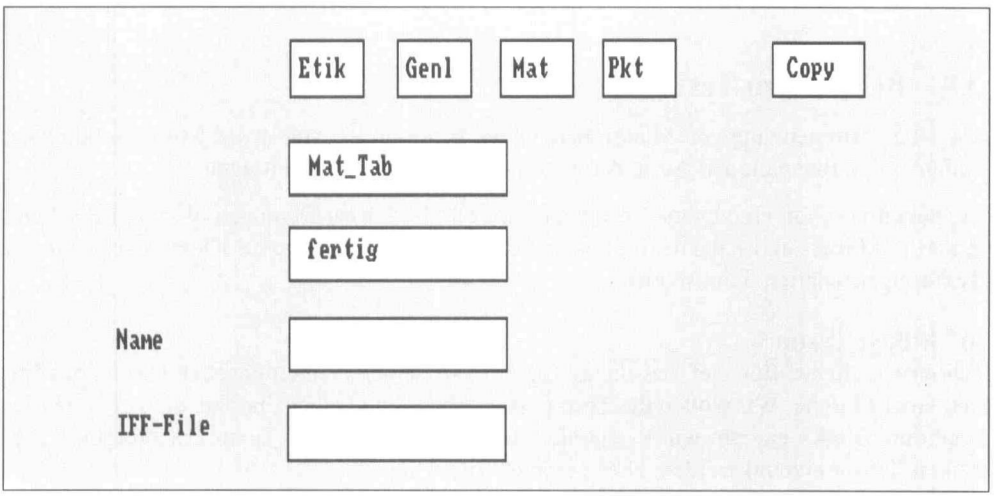


Bild 3.21: Die Gadgets bei »EditTex« und »NeuTex«

»Construct« aus der Datei die Größe des Bildes (Zeilen und Spalten) sowie die Anzahl der Farben. Diese drei Werte werden daraufhin im Listenfenster dargestellt.

IFF bedeutet »Interchange-File-Format«. Es ist ein auf dem Amiga standardisiertes Dateiformat. Es gibt für viele Anwendungen IFF-Standards, aber am bekanntesten ist das Format für Bilder. Der Ausdruck IFF-Bild steht immer für eine Datei, die im IFF-Format ein Bild gespeichert hat. Die meisten Malprogramme (auf jeden Fall alle kommerziellen) können IFF-Bilder erzeugen. Reflections kann fast alle IFF-Bilder verwenden, mit folgenden Einschränkungen:

- HAM-Bilder werden nicht korrekt benutzt (siehe auch Kapitel 5). Bei solchen Bildern hängt die Pixelfarbe meist von der Farbe des Nachbarpixels ab (und dessen Farbe wieder von dessen Nachbarpixel usw.). Die Bilder können zwar verwendet werden, aber das Ergebnis sieht bestimmt anders aus, als Sie es erwartet haben.
- Reflections unterstützt auch nicht den Extra-Halfbrite-Modus. Auch solche Bilder werden nicht korrekt bearbeitet. Sie können Bilder von Reflections als Texturen verwenden, dürfen sie dann aber mit »Show« (Kapitel 5) nicht als HAM-Bild speichern.

Die Anzahl der verwendeten Texturen ist begrenzt. Eine Begrenzung liegt bei »Beams«. Der kann nur so viele Texturdateien laden, wie Platz im Rechner frei ist, maximal aber 20 verschiedene Texturdateien.

Es können viele verschiedene Texturobjekte die gleiche Texturdatei benutzen. Die wird jeweils nur einmal geladen. Maximal 100 Texturobjekte sind möglich. Sie können also maximal an 100 Körper Texturen vergeben.

3.8 Beispiele zu Texturen

Da der Zusammenhang von Materialien, Oberflächen und Textur etwas unübersichtlich ist, sollen einige Beispiele zur Auflockerung und zum Verständnis beitragen:

Sie haben ja schon eine kleine Test-Textur im Farbbild 3 zur Demonstration von Flächen-, Rotationskörper- und Kugeltextur kennengelernt. Mit der Textur will ich einmal einige der Textureigenschaften durchspielen:

a) Etikett-Textur

Ein anschauliches Beispiel, das dieser Eigenschaft seinen Namen gegeben hat, ist das Etikett einer Flasche. Wir wollen die Test-Textur nehmen und einer Flasche, die zuvor mit der Funktion »Rotk« erzeugt wurde, zuteilen. Dazu muß zuvor eine Texturbeschreibung einer Etikett-Textur erzeugt werden. Dies geschieht mit »NeuTex«.

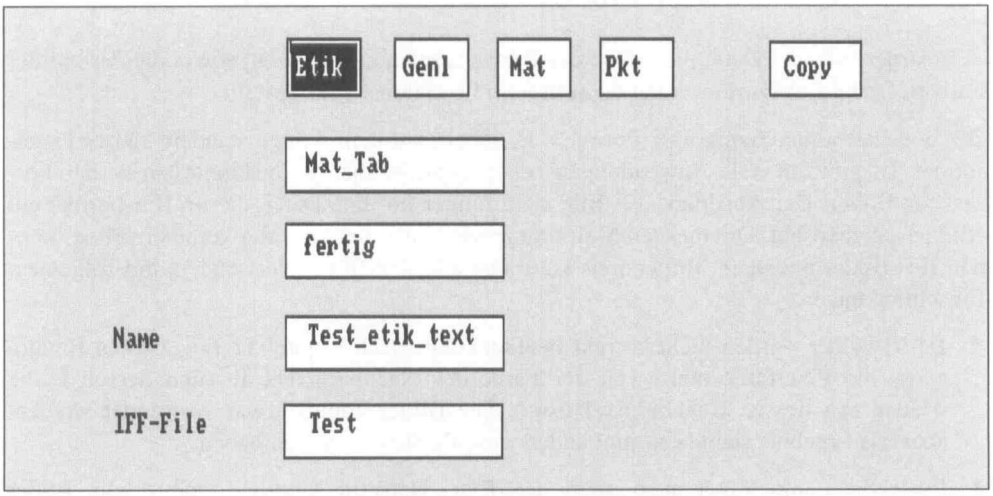


Bild 3.22: Die Gadgets von »NeuTex«

Das »Etik«-Feld muß angeklickt werden, um diese Textur als Etikett zu kennzeichnen. Dann werden Name und IFF-Dateiname festgelegt. Zusätzlich muß jetzt in der Material-

tabelle der Textur der erste Eintrag erfolgen. Das ist das Material, das das Etikett haben soll. Hier nehmen wir irgendein nicht durchsichtiges Material, etwa Blau-Metall. (Die Farbe von Blau-Metall spielt hier keine Rolle, denn sie wird ja durch die Farbe des Texturbildes ersetzt.) Damit sind wir aber noch nicht fertig. An Körper wird ja keine Textur, sondern ein Material vergeben. Es muß somit noch ein neues Material erzeugt werden, das die soeben geschaffene Textur enthält.

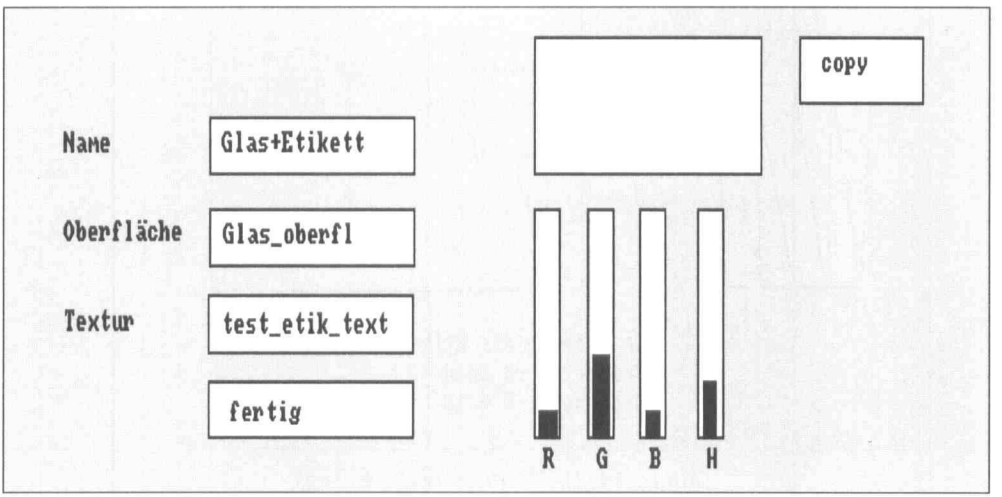


Bild 3.23: »Etik« im »NeuMat«-Requester

Wir geben dem Material den Namen »Glas+Etikett«. Als Oberfläche nehmen wir »Glas_oberfl«, denn die Flasche soll ja aus Glas sein. Nur das Etikett ist in dem Fall aus Metall. Als Farbe könnte man ein dunkles Grün nehmen. Als Textur wird jetzt selbstverständlich die soeben erzeugte Textur verwendet.

Nun kann der Flasche das Material »Glas+Etikett« zugeteilt werden; doch wir müssen auf die Orientierung der Flasche achten! Zuerst wird die Flasche so gedreht, daß sie aufrecht steht (mit dem Hals nach oben). Zur Sicherheit plotten Sie sie einmal von vorne. Danach wird im »Körper«-Menü die Funktion »Material« angeklickt, und dem Körper Flasche das Material »Glas+Etikett« zugeteilt.

Daraufhin fragt »Construct« nach der Texturart; wir wählen die Rotationskörpertextur. Diese muß so skaliert werden, daß das Etikett den richtigen Bereich auf der Flasche einnimmt. Bild 3.24 zeigt die Skalierung des Etiketts.

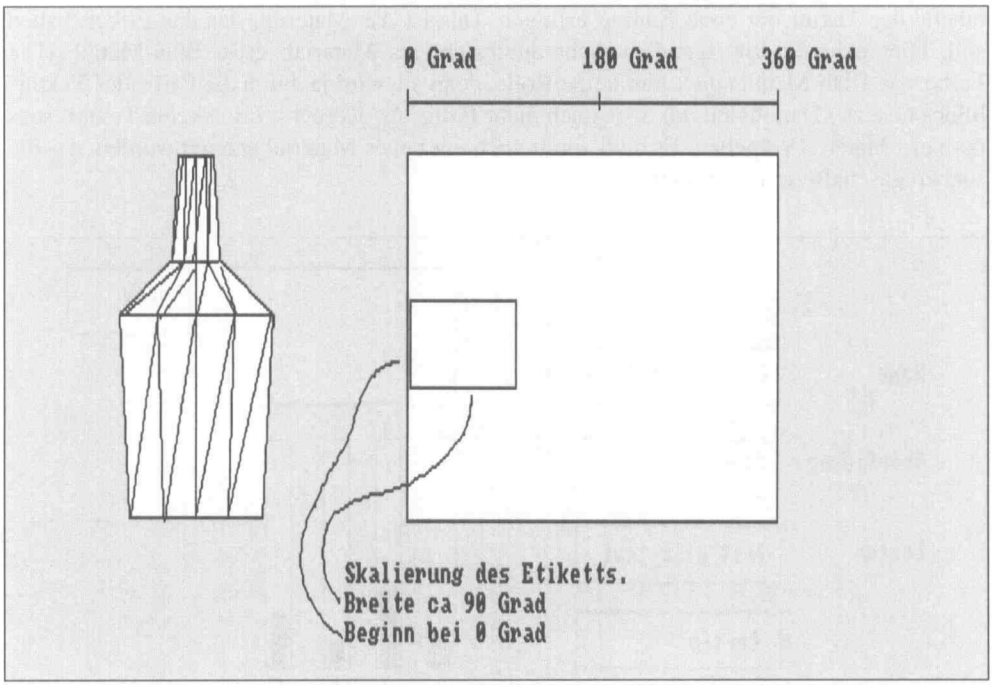


Bild 3.24: Die Skalierung des Etiketts

Die Textur wird so skaliert, daß sie, von der Höhe her betrachtet, etwa in der Mitte der Flasche liegt, in der Breite etwa 90 Grad bedeckt und bei 0 Grad beginnt. Das Ergebnis sehen wir in Farbbild 8.

b) Genlock-Textur

Genlock-Textur unterscheidet sich von normaler Textur nur in folgenden Punkten:

- Für alle Pixel (im IFF-Bild) mit Farbe 0 wird Farbe und Oberfläche des Grundmaterials verwendet.
- Für alle anderen Pixel gelten die Materialeigenschaften des ersten Materials der Materialtabelle der Textur (wie bei Etikett).

Wir nehmen wieder die Test-Textur. Erneut müssen wir zuerst eine neue Texturbeschreibung anlegen (mit »NeuTex«). Die wird mit dem »Genl«-Schalter gekennzeichnet, als Texturname wird zum Beispiel »Test-genl-text« angegeben.

Wie bei Etikett müssen wir auch hier den ersten Eintrag der Materialtabelle setzen, zum Beispiel wieder mit »Blau-Metall«. Danach wird ein neues Material benötigt. Nennen wir es »Genl_test_mat«. Als Oberfläche ordnen wir ihm »Glas_Oberfl« zu, wie der Flasche.

Alle Teile des Körpers, die von Farbe 0 des Texturmusters bedeckt sind, werden dadurch wie Glas behandelt, während die anderen eine metallische Oberfläche bekommen.

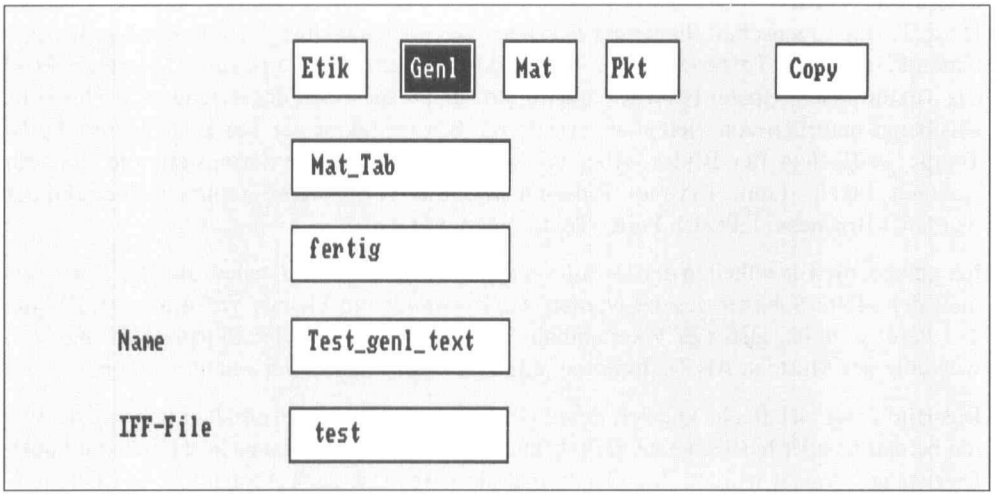


Bild 3.25: Genlock-Textur

In Farbbild 9 sehen Sie links eine Kugel mit einfacher Farb-Textur, während die rechte Kugel das neue Genlock-Material hat. Wie Sie sehen, sind bei der rechten Kugel einige Stellen durchsichtig. Dort hatte das Texturmuster Farbe 0, deshalb wurde an der Stelle das Grundmaterial mit »Glas_Oberfläche« verwendet.

c) Material-Textur

Dies ist ein weiterer Schritt gegenüber der »Genlock«-Textur. Dort hatten alle Pixel mit Farbe 0 ein eigenes Material (Grundfarbe des Materials und Oberfläche). Alle anderen Pixel des Textur-Bildes hatten die Materialeigenschaft, die als erste in der Materialtabelle der Textur steht. Bei Materialtextur kann jeder Farbe des Texturmusters ein eigenes Material zugeordnet werden. Wir nehmen wieder die Test-Textur. Mit »NeuTex« wird eine neue Textur erzeugt, bei der der »Mat«-Schalter gesetzt wird. Jetzt müssen die Materialien der Textur-Materialtabelle bestimmt werden. Dazu wird das »Mat-Tab«-Feld angeklickt.

Zuerst erscheinen im List-Fenster die bisherigen Materialien der Textur, verbunden mit der Frage, ob sie geändert werden sollen. Nach Anklicken von »ändern« klicken Sie nacheinander im Auswahlfenster die Materialien »Gold«, »Spiegel«, »Blau-Metall« und »Glas« an. (Das Textur-Bild »test« besteht aus vier Farben, daher können maximal vier Materialien angegeben werden.)

In Farbbild 10 sehen Sie zwei Kugeln. Die rechte hat das soeben vergebene Material, während der linken ein anderes Material zugeordnet ist, das ebenfalls eine Materialtextur ent-

hält. Dabei ist nur Farbe 0 auf »Blau-Metall« gesetzt, alle anderen Farben haben Material »nichts«, sind also durchsichtig.

d) Punkt-Textur

Diese Textur-Eigenschaft dient zum Erzeugen von weichen Übergängen zwischen benachbarten Pixeln eines Textur-Musters. Der Effekt ist allerdings nur erkennbar, wenn die Pixel des Texturmusters später beim Raytracing größer als die Pixel des erzeugten Bildes sind. Das hängt natürlich von vielem ab, wie Vergrößerungsfaktor der Kamera, Skalierung der Textur, Auflösung der Bildes. Aber vor allem bei kleinen Texturmustern mit nur sehr wenigen Pixeln (zum Beispiel Pinselmuster, die von Malprogrammen übernommen wurden, »Brushes«) läßt sich Punkt-Textur gut einsetzen.

Ich glaube, die Einzelheiten der Definition kann ich Ihnen jetzt ersparen. Bei der Definition muß der »Pkt«-Schalter gesetzt werden. Ich verwende ein kleines Texturmuster, das nur 2*2 Pixel groß ist, also vier Pixel enthält. Farbbild 11 enthält vier Beispiele für die Verwendung des Musters. Als Texturart ist jedesmal Flächentextur verwendet worden.

Das Bild zeigt vier flache Quader, deren Oberseite je eine Textur erhalten haben. Die oberen beiden Quader besitzen eine Farb-Textur. Beim rechten oberen ist zusätzlich die Punkt-Eigenschaft gesetzt worden. Man sieht, daß die Farben, je nach Abstand zu den Eckpunkten, ineinander übergehen. Noch interessanter wird es aber bei Material-Textur. Die unteren beiden Quader sind Beispiele dafür, der rechte wieder zusätzlich mit Punkt-Textur. Die Textur enthält statt vier Farben vier Materialien, nämlich »Blau-Metall«, »Messing«, »Hellblau-Matt« und »Glas«. Beim rechten Quader gehen nun nicht nur die Farben, sondern auch die übrigen Eigenschaften der Materialien ineinander über. Das kann man besonders bei der Glas-Eigenschaft erkennen. Zur Verdeutlichung haben die unteren beiden Quader an ihrer Unterseite noch die Test-Textur aus den vorherigen Beispielen. Diese kann man dort sehen, wo die Oberseite das Material Glas besitzt.

Am Ende dieses Kapitels möchte ich noch tabellarisch die verschiedenen Eigenschaften sowie ihre Kombinationen erläutern. Die einzelnen Eigenschaften sind in der Tabelle wie folgt gekennzeichnet:

E=Etikett
G=Genlock
M=Material
P=Punkt
MT=Materialtabelle der Textur
GM=Grundmaterial
GF=Grundfarbe
TF=Texturfarbe = Farbe des getroffenen Pixels der Textur

Wenn die Farbe nicht extra aufgeführt wird, ist die Grundfarbe des Materials gemeint (ein Material besteht aus Oberfläche und Grundfarbe).

Die Kombination von Genlock- (G) und Material-Textur (M) führt bei der Frage, welches Material für Farbe 0 verwendet werden soll, zum Konflikt. Ich habe mich hier für die Material-Textur entschieden, das heißt, bei Material-Textur wird die Genlock-Eigenschaft nicht beachtet.

1) Normale Farbtextur ohne besondere Eigenschaften
Material: GM
Farbe: TF

2) E
Innerhalb Skalierung:
Material: MT(0)
Farbe: TF

Außerhalb Skalierung:
Material: GM
Farbe: GF

3) G
Material: Falls TF=0 GM, sonst MT(0)
Farbe: TF, falls TF=0 dann GF

4) M
Material: MT(TF)

5) P
Material: GM
Farbe: Interpoliert zwischen TF und benachbarten Pixeln der Textur.

6) E+G
Innerhalb Skalierung:
Material: Falls TF=0 dann GM, sonst MT(0)
Farbe: Falls TF=0 dann GF, sonst TF

Außerhalb Skalierung:
Material: GM
Farbe: GF

7) E+M
Innerhalb Skalierung:
Material: MT(TF)

Außerhalb Skalierung:
Material: GM
Farbe:GF

8) E+P
Innerhalb Skalierung:
Material: MT(0)
Farbe: Interpoliert zwischen TF und benachbarten Pixeln der Textur.

Außerhalb Skalierung:
Material: GM
Farbe: GF

9) G+M
Genau wie 4)

10) G+P
Material: Falls TF=0 GM, sonst MT(0)
Farbe: Interpoliert zwischen TF (falls TF>0) oder GF (falls TF=0) und benachbarten Pixeln der Textur.

11) M+P
Material: Interpoliert zwischen MT(TF) und MT (benachbarte Pixel).

12) E+G+M
wie 7) E+M

13) E+M+P
Innerhalb Skalierung:
Material: wie 11)

Außerhalb Skalierung:
Material: GM
Farbe: GF

14) G+M+P
wie 11)

15) E+G+M+P
wie 13)

3.9 Lichtquellen

Bis jetzt haben wir die Geometrie einer Szene sowie die Materialeigenschaften von Objekten behandelt. Für das Raytracing fehlen allerdings noch Angaben über die Lichtverhältnisse.

In Reflections kann Licht entweder von Lichtquellen kommen oder, als Umgebungslicht, von überall her. Die Position der Lichtquellen wird beim Plotten von Körpern durch weiße Kreise angedeutet, sofern sie im Darstellungsbereich des Plotfensters zu sehen sind. Alle Funktionen hierzu sind im Menü »Licht« untergebracht.

Zu Beginn gibt es keine Lichtquellen. Sobald Sie Lichtquellen definiert haben, werden sie als weiße Kreise beim Plotten mitgezeichnet, wenn sie im Bereich des Bildschirms sind. So können Lichtquellen genau lokalisiert werden.

NeuLicht
»NeuLicht« dient zur Definition einer neuen Lichtquelle. Lichtquellen haben eine Farbe und eine Position. Bei Anwahl von »NeuLicht« ermittelt »Construct« zuerst einen Quader, der alle Objekte einschließt. Dann setzt es die neue Lichtquelle in die Mitte der oberen Seite des Quaders. In dieser Position werden alle Objekte von oben beleuchtet. Anschließend fragt »Construct« nach der Farbe der Lichtquelle. Im Auswahlfenster erscheinen darauf alle Materialien. »Construct« nimmt die Grundfarbe des angeklickten Materials. Klicken Sie mit rechts oder wählen Sie kein Material, wird als Lichtfarbe Weiß genommen. Danach möchte »Construct« noch die Stärke der Lichtquelle wissen. Sie können die mit dem Proportionalregler im Bereich von 0.1 bis 5.0 frei wählen. Bei Stärke 1.0 bleibt die Farbe unverändert. Bei zu starkem Licht tendieren beleuchtete Objekte zu weißer Farbe. Bei zu schwachem Licht kann es leicht passieren, daß man die Wirkung der Lichtquelle im Bild nicht mehr sieht. Es können maximal fünf Lichtquellen definiert werden.

Licht_Farbe
Hiermit kann Farbe und Stärke einer Lichtquelle geändert werden. Zu Beginn muß die Lichtquelle bestimmt werden, deren Farbe geändert werden soll. Dies geschieht durch Bewegen des Mauszeigers in die Nähe der Lichtquelle und Klicken mit der linken Maustaste. Falls die Lichtquelle, die Sie ändern wollen, nicht im Plotfenster als weißer Kreis gezeichnet ist, ist der Maßstab zu klein, und Sie müssen zuvor mit der Funktion »Bild*2« im »Plotten«-Menü den Maßstab so lange vergrößern, bis die Lichtquelle erscheint. Sobald eine Lichtquelle bestimmt wurde, zeigt »Construct« im Listfenster die Farbe der Lichtquelle als numerische Werte an und fragt, ob sie geändert werden soll (Rot-, Grün- und Blauwerte von jeweils 1.0 entsprechen Weiß). Falls ja, wird wieder, wie bei »NeuLicht«, die Farbe eines von Ihnen gewählten Materials genommen und die Stärke mit dem Proportionalregler angegeben.

Licht_Pos
Hier wird die Position einer Lichtquelle geändert. Wie in der Funktion »Lichtfarbe« zeigen Sie mit dem Mauszeiger, welche Lichtquelle geändert werden soll. Anschließend bewegen

Sie den Mauszeiger an die neue Position der Lichtquelle und klicken mit der linken Maustaste. Die Lichtquelle wird dann an der neuen Position gezeichnet. Schauen Sie sich anschließend die Szene von einer anderen Position aus an und wiederholen Sie das ganze gegebenenfalls, da eine Veränderung der Position nur in zwei Richtungen auf einmal möglich ist.

Lichtweg
Hiermit können Sie eine Lichtquelle löschen. Klicken Sie wie bei »Lichtfarbe« oder »LichtPos« die entsprechende Lichtquelle an. Die wird dann gelöscht.

Hintergrund
Mit »Hintergrund« wird die Hintergrundfarbe angegeben. Wie bei »Lichtfarbe« geschieht das über die Materialtabelle. Es wird die Grundfarbe des gewählten Materials als Hintergrundfarbe des Raytrace-Bildes genommen. Trifft beim Raytracing ein Strahl kein Objekt, so liefert er als Farbe die Hintergrundfarbe zurück.

Allerdings kann auch eine Textur als Hintergrund benutzt werden. Das geschieht, wenn Sie als Material ein Texturmaterial (Rot im Auswahlfenster) wählen. In dem Fall ist der weitere Ablauf ähnlich wie bei Vergabe eines Texturmaterials an einen Körper. Sie müssen die Texturart angeben, wobei hier allerdings nur Flächentextur und Kugeltextur möglich ist. Bild 3.26 verdeutlicht Flächentextur als Hintergrund.

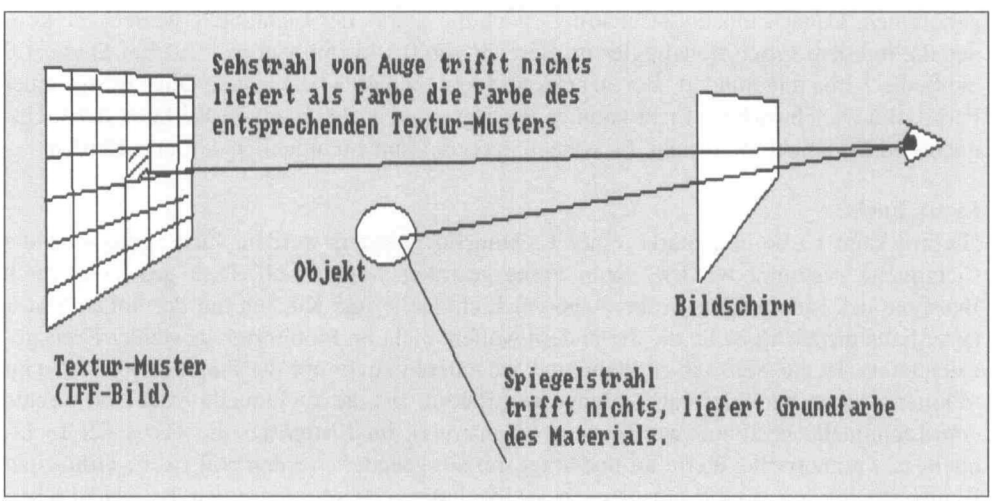


Bild 3.26: Flächentextur als Hintergrund

Bei Flächentextur bekommen die Strahlen, die vom Auge ausgehen und nichts treffen, die entsprechende Farbe der Hintergrundtextur. Reflektierte Strahlen, die nichts treffen,

bekommen als Farbe allerdings die Materialgrundfarbe und haben mit der Textur nichts zu tun.

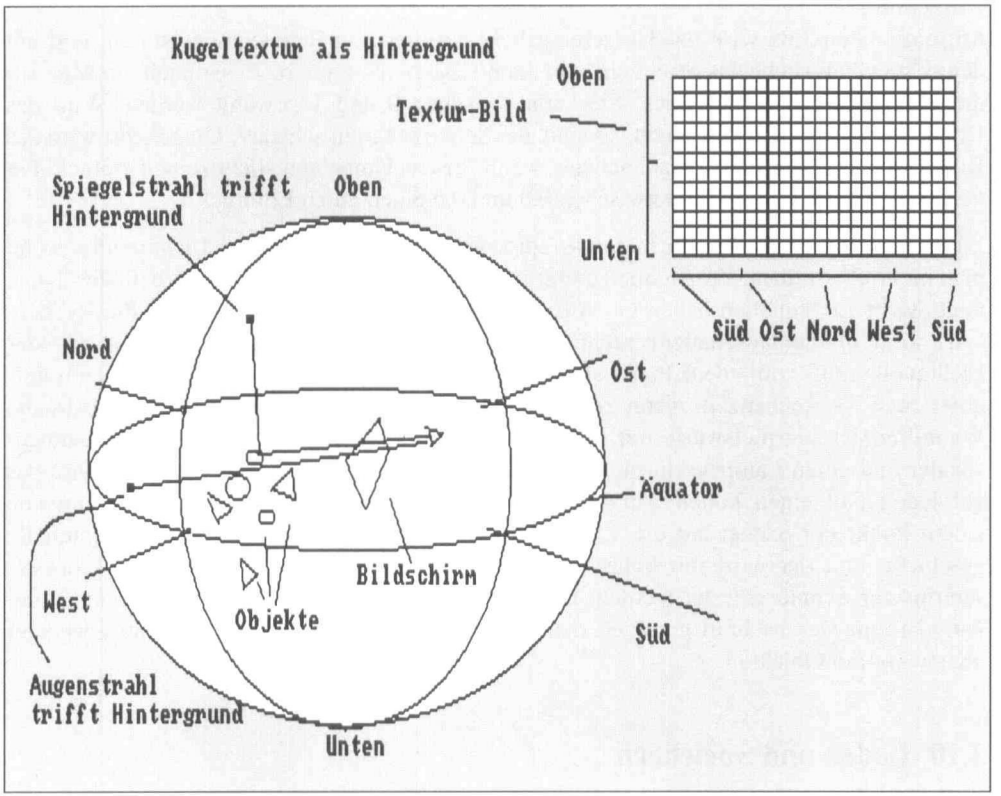


Bild 3.27: Kugeltextur als Hintergrund

Bei Kugeltextur ist der Hintergrund stark von der Kameraposition und der Blickrichtung abhängig. Man sieht immer nur einen kleinen Ausschnitt der umgebenden Kugel, also auch nur einen kleinen Teil des Texturmusters. Die Pixel des Texturbildes sind deshalb sehr stark vergrößert. Dieser Effekt kann sehr störend sein. Es hilft etwas, wenn Sie bei der Funktion »Kam.Beweg« die Vergrößerung recht klein wählen und stattdessen sich mit der Kamera vor- bzw. zurückbewegen, um die Größe der Objekte auf dem Bildschirm zu kontrollieren.

Die Skalierung und Positionierung des Texturmusters erfolgt genauso wie bei der Texturvergabe an einen Körper.

Es bleibt noch zu erwähnen, daß bei Hintergrundtextur keine Materialtextur verwendet wird; das heißt: Egal, ob als Textureigenschaft Materialtextur vereinbart ist oder nicht, es wird nur die Farbe des Texturbildes verwendet. Dies liegt daran, daß der Hintergrund nicht

an der Beleuchtungsformel teilnimmt (er liegt ja außerhalb des Szene) und daher nur als Farbmuster interpretiert werden darf.

Umgebung
Mit dieser Funktion wird das Umgebungslicht definiert. Im Beleuchtungsmodell wird mit dem Umgebungslicht das Streulicht von allen Objekten simuliert. Es ist auch ein Maß für die Grundhelligkeit des Bildes. Sie kann zwischen 0 und 1 gewählt werden. Wird das Umgebungslicht gleich 0 gesetzt, so sind alle Schlagschatten schwarz. Umgekehrt wird das Bild als Ganzes ziemlich »ausbleichen«, wenn Sie das Umgebungslicht zu hell wählen. Für normale Szenen liefern Werte zwischen 0.3 und 0.5 einen guten Eindruck.

Zu den Lichtquellen ist noch folgendes zu sagen: Bei der Anzahl der Lichtquellen sollte man nicht übertreiben. Zu Anfang genügt immer eine Lichtquelle, die außerhalb der Szene liegt. Mehr Lichtquellen haben einen Einfluß aufs Raytracing (vor allem auf die Rechenzeit), wenn die Schlagschattenberechnung (siehe Kapitel 4) eingeschaltet ist, denn zu jeder Lichtquelle muß von jedem Punkt aus ein Strahl geschickt werden. Stark erhöht sich mitunter auch die Rechenzeit, wenn Sie eine Lichtquelle in die Szene setzen statt außerhalb. Wenn Sie sich beispielsweise mit »Construct« eine Lampe erzeugen, also einen Lampenständer und einen Lampenschirm, und in den Schirm eine Lichtquelle setzen, so wird das auf jeden Fall einen hohen Aufwand bei der Schattenberechnung bedeuten; denn von jedem Punkt der Szene, auf die ein Strahl stößt, wird ein Schattenstrahl zur Lichtquelle geschickt, und der wird mit hoher Wahrscheinlichkeit mit den Objekten des Lampenschirms auf Schnitt getestet werden. Liegt der Lichtquelle außerhalb der Szene, so ist der Aufwand im Durchschnitt geringer, denn die Schattenstrahlen haben dann meist weniger »Kontakt« mit Objekten.

3.10 Laden und Speichern

Nun kommt der Menüpunkt »Projekt« zur Sprache. Er beinhaltet alle Funktionen, um Szenen, Materialien, Lichtquellen und Kameradaten auf Dateien zu speichern oder von Dateien zu laden, und schließlich kann man hier das Programm verlassen.

Zur Festlegung der Datei-Namen erscheint im Plotfenster die Datei-Auswahlbox. Mit ihr können Ordner festgelegt oder angeschaut werden, Dateien ausgewählt und Namen eingegeben werden. Bild 3.28 zeigt die Datei-Auswahlbox.

In dem Requester kann folgendes festgelegt werden:

Oben finden Sie einige Schalter:

ok: Wenn Sie mit dem Dateinamen zufrieden sind, klicken Sie »ok« an.

abbr: Zum Abbrechen der ganzen Prozedur.

Take: Im aktuellen Ordner werden bei Anklicken von »Take« alle Dateinamen gelesen und im Auswahlfenster dargestellt. Normale Dateien sind blau, Ordner rot gezeichnet. Mit der Maus können Sie daraus eine Datei oder einen Ordner auswählen.

df0:
df1: Das jeweilige Laufwerk als Ordner ansprechen.
dh0:
ram:

Unter den Schaltern sind zwei Felder, mit denen Sie den Ordner und den Dateinamen festlegen können.

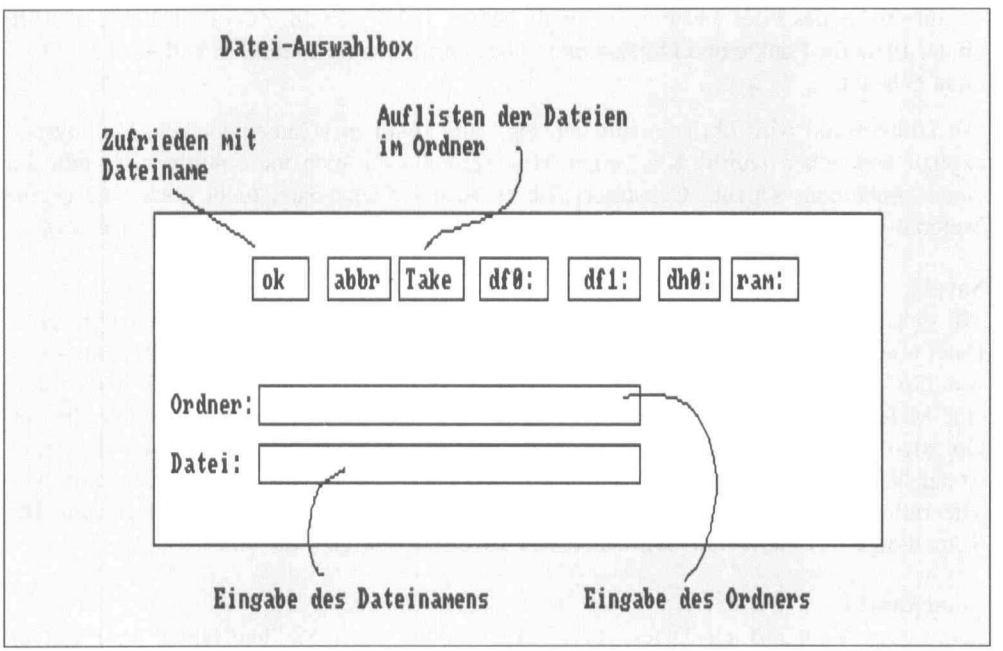


Bild 3.28: Datei-Auswahlbox

Ordner: Wenn Sie das Feld anklicken, erscheint das Fragefenster. Dort können Sie den Namen des Ordners, in dem die gewünschte Datei liegt, eintippen.

Datei: Beim Anklicken dieses Feldes können Sie im Fragefenster den Dateinamen eintippen.

»Construct« kennt drei Arten von Dateien:

- *Szenendateien.* Sie enthalten Punkte, Objekte, Körper, Texturobjekte und Hintergrund einer Szene.
- *Kamera/Licht-Dateien.* In ihnen sind alle Angaben zur Kameraposition, Blickrichtung, Vergrößerung etc., die Positionen und Farben der Lichtquellen sowie die Stärke des Umgebungslichtes gespeichert.
- *Materialdateien.* Darin finden sich Textur-, Oberflächen- und Materialbeschreibungen.

LoadSz
Hiermit wird eine Szenendatei geladen. Mit Hilfe der Datei-Auswahlbox müssen Sie den Namen der Szenendatei angeben. »Construct« lädt dann die Punkte, Objekte, Körper und Texturobjekte des Files und fügt sie an die bereits definierten an. Zuvor wird aber geprüft, ob der Platz für Punkte und Objekte nicht überschritten wird. In diesem Fall wird die Datei nicht geladen.

Der Hintergrund wird übrigens nur geladen, falls zuvor in »Construct« kein Hintergrund explizit angegeben wurde. Klar, einen Hintergrund kann man nicht anfügen, es gibt nur einen. Außerdem schaut »Construct«, ob es zu der Szenendatei nicht noch eine eigene Materialdatei gibt; wenn ja, wird diese automatisch dazugeladen (siehe auch »SaveSz«).

SaveSz
Mit dieser Funktion wird die gesamte Szene, die »Construct« gespeichert hat, in einer Datei abgelegt. Zur Szene gehört auch der Hintergrund. Wieder erscheint die Datei-Auswahlbox zur Festlegung des Dateinamens sowie des Ordners. Zusätzlich legt »Construct« eine Materialdatei an, die nur die Materialien, Oberflächen und Texturen enthält, die von Objekten der Szene belegt sind. Diese Materialdatei hat den Namen »Szene.mat« (»Szene« ist der Name der Szenendatei). Sie wurde eingerichtet, da sich mit der Zeit doch sehr viele Materialien ansammeln, von denen in einer konkreten Szene nur wenige belegt sind. Die Szenen-eigene Materialdatei wird auch von »Beams« gelesen.

LoadKamLi
Laden einer Kamera/Licht-Datei. Mit der Auswahlbox geben Sie den Namen der Datei an. »Construct« lädt daraufhin die entsprechenden Kamera- und Lichtdaten. Falls Sie vorher eine Kameraposition definiert hatten, ist sie jetzt überschrieben. Genauso ist es bei den Lichtquellen. Mit dem Laden einer Kamera/Licht-Datei sind Kamera und Lichtquellen (auch Umgebungslicht) jedesmal neu definiert.

SaveKamLi
Zum Sichern von Kamera und Lichtquellen. Dazu müssen Sie allerdings zuvor Lichtquellen definiert haben. »Construct« fügt an den Dateinamen, den Sie mit Hilfe der Datei-Auswahlbox angeben, automatisch den Suffix ».kam« an, um die Datei als Kamera/Licht-Datei zu kennzeichnen. Für »Construct« brauchen Sie diese Datei nicht unbedingt. Aber

der Raytracer »Beams« ist darauf angewiesen. Spätestens wenn Sie »raytracen« wollen, müssen Sie diese Datei speichern.

LoadMat
Zum Laden einer Materialdatei. Die Materialdateien, die mit »LoadSz« dazugeladen werden, enthalten nur die Materialien, die auch von Objekten der geladenen Szenendatei belegt sind. »LoadMat« wird eher dazu verwendet, Materialdateien zu laden, die nicht speziellen Szenen zugeordnet sind. Beim Laden von Materialdateien sollten die Namen von Materialien, Oberflächen und Texturen beachtet werden. »Construct« unterscheidet ausschließlich nach dem Namen. Wenn Sie ein Material »Blau-Metall« bereits definiert haben, und ein solches kommt auch in einer Materialdatei vor, die geladen wird, so geht »Construct« davon aus, daß beide gleich sind und überliest das zweite »Blau-Metall«. Das gleiche gilt übrigens auch für die Materialdateien, die den Szenendateien zugeordnet sind. Wenn Sie mehrere Szenen laden und in jeder Szene kommen Objekte mit zum Beispiel dem Material »Blau-Metall« vor, so wird das nur beim ersten Mal geladen.

Dies kann auch von Nachteil sein, wenn Sie für eine konkrete Szene einmal ein Material ein wenig ändern, den Namen aber belassen. Wenn Sie die Szene später wieder laden, kann es vorkommen, daß die Änderung verlorengeht, da »Construct« nur auf den Namen des Materials schaut.

SaveMat
Hiermit werden alle gespeicherten Materialien, Oberflächen und Texturen auf Datei abgelegt.

Progr.Ende
Hier befindet sich der Ausgang aus »Construct«. Es wird sicherheitshalber noch einmal nachgefragt, ob Sie auch wirklich abbrechen wollen. (Haben Sie auch alles abgespeichert, was eventuell mühsam erzeugt wurde?) Falls Sie nur kurz mal etwas anderes als »Construct« benutzen wollen und der Speicher ausreicht, dann lohnt es sich nicht abzubrechen. In dem Fall ist es schneller, den Schirm von »Construct« mit dem Tiefen-Gadget in den Hintergrund zu klicken; Sie können solange mit der Workbench weiterarbeiten. Um wieder zu »Construct« zu kommen, muß beim Workbench-Schirm wieder das Tiefen-Gadget geklickt werden.

3.11 Beispiele

In diesem letzten Abschnitt zu »Construct« möchte ich Ihnen eine lockere Sammlung von Tips geben, die Ihnen vielleicht als kleine Starthilfe nützlich ist. Die Sammlung kann sicher nie vollständig sein, ich selbst finde noch oft neue Möglichkeiten, die mir vorher nie in den Sinn kamen. Aber einige Beispiele sollten wohl zum Verständnis des Zusammenwirkens der verschiedenen Funktionen von »Construct« beitragen.

Sie sollten die Beispiele direkt mit »Construct« durchspielen; durch Praxis lernt man sehr viel, und so werden vielleicht schon viele Fragen beantwortet, die Ihnen beim Durchlesen in den Sinn gekommen sind. Auf die genaue Handhabung der einzelnen Funktionen wird hier aber nicht mehr eingegangen. Wenn hierzu Fragen auftauchen, schlagen Sie am besten in den zurückliegenden Kapiteln nach.

Ein gewisses Problem entsteht durch die fehlenden Größenangaben bei »Construct«. Es werden Körper erzeugt, aber man hat keinen Bezugspunkt, anhand dessen man die Größe beurteilen kann. In »Construct« können Körper nur in Bezug zu anderen Körpern beurteilt werden.

Ein Grundprinzip bei »Construct« ist das Bausteinprinzip. Komplexe Körper sind aus Bausteinen zusammengesetzt, nämlich aus anderen Körpern.

Wie sieht das in der Praxis aus? Wie setzt man Körper zusammen, von denen man weder die Größe noch die gegenseitige Lage kennt? Wie wird beispielsweise ein Glas, das zuvor mit der »Rotk«-Funktion erzeugt wurde, auf einen Würfel gesetzt – ein recht einfaches Beispiel?

1) Setzen eines Glases auf einen Würfel
Wir erzeugen zuvor ein Glas mit der »Rotk«-Funktion und einen Würfel mit der »Quader«-Funktion (beide im »Tools«-Menü).

1. Schritt: Erzeugen eines Körpers, der sowohl »Glas« als auch »Würfel« enthält.

Dies geschieht mit »AddNeu« im »Körper«-Menü. Wir nennen den neuen Körper »Glas+Würfel«.

2. Schritt: Orientieren über die Lage von »Glas+Würfel«.

Mit »Fenster« (»Plotten«-Menü) stellen wir das Plotfenster auf den neuen Körper ein und plotten ihn von mehreren Seiten, um zu sehen, wie »Glas« und »Würfel« zueinander liegen (zum Beispiel mit »Vorne«, »Links« und »Oben« im »Plotten«-Menü).

3. Schritt: »Glas« auf »Würfel« zu bewegen (oder umgekehrt) und zugleich die Größe verändern, so daß die richtigen Größenverhältnisse zustandekommen.

Dies geschieht mit der Funktion »Verändern« (»Geomet«-Menü). Danach mit »Fenster« erneut das Plotfenster auf »Glas+Würfel« einstellen, da die Körper näher zusammengerückt sind. So kann der Maßstab verbessert werden.

4. Schritt: »Glas« so drehen, daß es aufrecht steht.

Wir plotten »Glas+Würfel« von vorne (mit »Vorne« im »Plotten«-Menü) und drehen dann »Glas« betragsmäßig um 90 Grad (»Drehen« im »Geomet«-Menü). Danach bewegen wir »Glas« so, daß es auf der Oberkante von »Würfel« steht. Eventuell muß danach »Glas« erneut bewegt werden. Um festzustellen, ob »Glas« auch wirklich auf »Würfel« steht, schauen wir das ganze von links an (Funktion »Links«) und bewegen »Glas« erneut, falls notwendig.

5. Schritt: Feinabstimmung.

Falls »Würfel« sehr viel größer als »Glas« ist, so kann es noch zu Ungenauigkeiten kommen, da »Glas« auf dem Bildschirm zu klein ist. Mit »Fenster« (»Plotten«-Menü) stellen wir den Maßstab auf »Glas« ein. Jetzt erscheint »Glas« ganz groß im Plot-Fenster, aber »Würfel« ist verschwunden. Mit »Plotkörper« (»Plotten«-Menü) sagen wir »Construct«, daß wir erneut »Glas+Würfel« geplottet haben wollen. Der Maßstab bleibt aber auf »Glas« eingestellt. Wir plotten »Glas+Würfel« erneut und können jetzt genau sehen, ob »Glas« genau auf »Würfel« steht. Gegebenenfalls bewegen wir »Glas« erneut, bis die Lage stimmt.

Das Glas fühlt sich jetzt auf dem Würfel vielleicht etwas einsam, außerdem ist der Würfel leer. Wir erzeugen weitere Körper (mit Funktionen aus dem »Tools«-Menü) und wollen sie auf den Würfel setzen.

2) Erweitern eines Körpers um neue Körper
Das Grundprinzip bleibt das gleiche wie im ersten Beispiel. Wir wissen kaum etwas über die Lage der neuen Körper und müssen sie daher zusammen mit »Würfel« auf dem Bildschirm sehen, aufeinanderzubewegen und jeweils den Maßstab erneuern.

Zuvor geben wir »Glas+Würfel« einen neuen Namen, zum Beispiel »Dinger_auf_Würfel«. Dies geschieht einfach mit »AddNeu«, wobei als einziger Körper »Glas+Würfel« angeklickt wird. Jetzt erzeugen wir einen neuen Körper, zum Beispiel eine Kugel.

1. Schritt: Zusammenfügen des neuen Körpers mit »Dinger_auf_Würfel«.

Diesmal benutzen wir statt »AddNeu« »Add« (im »Körper«-Menü), denn es soll ja kein neuer Körper erzeugt werden, sondern »Dinger_auf_Würfel« soll erweitert werden.

Schritte 2–5 wie oben.

So wird mit jedem neuen Körper verfahren, der auf den Würfel gesetzt werden soll.

Zwischendurch schauen wir uns die Szene einmal perspektivisch an. Mit Hilfe der »Kamera«-Funktion (»Plotten«-Menü) bestimmen wir einen geeigneten Beobachter-Punkt.

Arbeiten Sie das zweite Beispiel einmal für mehrere Körper durch, zum Beispiel für Zylinder, Quader, Kegel, Kugeln, ein 3D_Polygon etc. Aber übertreiben Sie es zu Anfang nicht mit der Objektanzahl. Vor allem mit der »Rotk«-Funktion kann man rasch sehr viele Objekte erzeugen.

So sammelt sich auf dem Würfel eine Anzahl von verschiedenen Körpern an. Angenommen, das Glas aus dem ersten Beispiel gefiele uns sehr gut. Davon wollten wir mehr haben:

3) Verdoppelung von Körpern
Wir stellen den Maßstab des Plot_Fensters auf »Dinger_auf_Würfel« ein und plotten den Körper von oben.

1. Schritt: Verdoppeln von »Glas« (mit »Doppeln« im »Körper«-Menü).

Wir nennen den neuen Körper »Glas1«. Aber irgendeine Wirkung ist nicht zu sehen, »Glas1« ist ja noch an der gleichen Stelle wie »Glas«.

2. Schritt: Wir bewegen »Glas1« etwas zur Seite (wo noch Platz auf dem Würfel ist). Jetzt ist »Glas« verschwunden. Beim Löschen der Zeichnung von »Glas1« wurde ja »Glas« auch gelöscht. Außerdem fügen wir mit »Add« »Glas1« zu »Dinger_auf_Würfel« dazu.

3. Schritt: Neuplotten von »Dinger_auf_Würfel«.

Das Prinzip des Zusammenfügens von Körpern sollte klargeworden sein.

Irgendwann sollten Sie übrigens den Körpern auch Materialeigenschaften geben. Wenn Sie das vergessen, haben die Objekte alle das »nichts«-Material und sind beim Raytracing nicht zu sehen. Am besten laden Sie gleich nach dem Start von »Construct« die Standard-Materialdatei »Reflections/mat/materialien« oder eine, die Sie sich selbst definiert haben. Im Prinzip ist es egal, wann Sie den Körpern Materialeigenschaften geben. Das kann auch erst ganz zum Schluß erfolgen. Nur zum Schluß kann es sein, daß Sie schon wieder einige Körperbeschreibungen gelöscht haben, und müssen nun mühsam mit »Define« Ihre Körper wieder zusammensuchen, um ihnen Materialien zu geben. Texturmaterialien würde ich aber sofort vergeben, wenn ein Körper geschaffen wurde, und zwar aus folgendem Grund: Zu Beginn ist die Orientierung der Textur (egal, welche Texturart gewählt wird) immer in irgendeiner Weise achsenparallel. Wenn ein Körper neu erzeugt ist, trifft das auch noch auf ihn zu. Später, wenn er gedreht wurde, wäre die Textur relativ schief auf ihm, jedenfalls ist es dann schwer abzuschätzen, wie das aussieht. Also: zuerst Texturmaterial vergeben und dann den Körper drehen.

Wenn Sie sich einen Körper geschaffen haben, den Sie auch noch in anderen Szenen verwenden wollen, sollten Sie sich den jetzt unter einem Dateinamen speichern. Zuvor sollten Sie aber alle Körper löschen, die zwar zur Erstellung notwendig waren, jetzt aber nicht mehr gebraucht werden. Andernfalls haben Sie, wenn Sie irgendwann die Datei wieder laden, eine Unzahl von Körpern, mit der Sie nichts anzufangen wissen.

Jetzt wieder etwas Konkretes. Es ist oft schwierig, Körper aneinanderzufügen, die nicht so schön symmetrisch wie ein Würfel sind oder nicht achsenparallel liegen.

4) Anfügen eines Weinglases an eine Kugel

Sie haben eine große Kugel erzeugt und ein Weinglas. Die Kugel ist sehr viel größer als das Weinglas. Das Weinglas ist zudem nicht achsenparallel, sondern schief.

1. Schritt: Definieren eines Körpers »Kugel+Glas« mit »AddNeu«.

Der Maßstab wird mit Fenster auf den Körper eingestellt. Dann wird das Weinglas zur Kugel hinbewegt, so gut es eben geht. (Weil die Kugel so groß ist, ist das Weinglas auf dem Bildschirm sehr klein und nicht gut zu erkennen.) Dies geschieht von mehreren Seiten, bis wir sicher sind, daß das Glas zumindest ungefähr auf der Kugeloberfläche liegt.

2. Schritt: Einstellen des Maßstabs auf das Weinglas.

Der Plotkörper wird aber mit »Plotkrp« wieder auf »Kugel+Glas« gesetzt. Wenn wir erneut plotten, füllt das Weinglas den Schirm, während die Kugeloberfläche nur als Kreissegment zu sehen ist. Falls die Kugeloberfläche nicht gezeichnet wird, ist sie noch zu weit entfernt. Mit »Bild*2« vergrößern wir dann den Maßstab, bis sie zu erkennen ist.

3. Schritt: Jetzt kann das Weinglas genau auf die Oberfläche positioniert werden. Außerdem wird es noch so gedreht, daß es genau senkrecht auf der Kugeloberfläche steht. Das wird jedesmal von mindestens zwei Seiten begutachtet und eventuell wiederholt, bis man zufrieden ist.

Eine Variation der letzten Übung wäre das Aneinanderfügen zweier sehr großer Körper. Der erste Schritt bleibt gleich. Wir plotten dann den gemeinsamen Körper mit »Hidd_2« und definieren uns einen weiteren Körper mit »define«, der alle Objekte der Kontaktstelle enthält. Auf diesen Körper stellen wir den Maßstab ein. Der Plotkörper bleibt aber auf den Körper eingestellt, der beide großen Körper enthält. Jetzt können wir die Körper ganz fein aufeinanderzubewegen.

Sie sollten sich also den Maßstab immer so einstellen, daß Sie den Raumbereich, der für eine Operation wichtig ist, immer so groß wie möglich darstellen. Den Maßstab richten Sie mit »Fenster« ein. Falls kein Körper existiert, der nur die Objekte in dem Bereich enthält, schaffen Sie sich einen mit »Define«.

Beams

4.1 Einleitung

»Beams« ist das Raytracing-Programm und damit der Kern des Reflections-Systems. Sie haben zwar als Benutzer nicht so viel mit »Beams« zu tun (den meisten »Kontakt« haben Sie mit »Construct«), der Amiga dafür um so mehr, denn »Beams« ist ein unheimlicher »Rechenzeit-Fresser«. Die Rechenzeit ist stark von den Gegebenheiten in der Szene und der gewünschten Bildqualität und -größe abhängig (in Kapitel 7 wird genauer darauf eingegangen). Die Rechenzeiten streuen gewaltig von wenigen Minuten bei kleinen Bildern und wenig Effekten bis zu einer ganzen Nacht bei einem vollen Bild mit ordentlich Spiegelung, Brechung, Texturen und Antialias-Berechnung.

Der Durchschnitt liegt nach meiner Erfahrung bei einer Lo-Res-Auflösung (siehe Kap. 5, »Show«) und etwa 1000 Objekten zwischen zwei und fünf Stunden, immer noch ein Zeitraum, der viel zu lange ist, um auf das Ergebnis zu warten oder mal eine Tasse Kaffee zu trinken.

Der Amiga hat aber glücklicherweise ein Multitasking-Betriebssystem, und das ist ein Segen für jemanden, der zwar Raytracing betreiben will, aber nebenbei noch etwas anderes auf dem Rechner zu tun hat.

Dabei wird die Tatsache ausgenutzt, daß bei normalen Anwendungen der Rechner kaum Arbeit hat. Viele Programme kommunizieren stark mit dem Benutzer, das heißt, sie müssen meistens auf irgendwelche Eingaben warten. Da der Mensch aber vergleichsweise langsam ist (schon der Zeitraum zwischen zwei Tastenanschlägen beim Eintippen eines Textes ist für den Rechner eine halbe Ewigkeit), ist es beim Amiga inzwischen Standard, daß sich ein Programm, wenn es auf irgend etwas wartet, sozusagen »zur Ruhe legt«. Es ist dann in einem Wartezustand, bei dem der zentrale Prozessor (CPU) nicht mehr belastet wird. Erst wenn ein bestimmtes Ereignis eingetreten ist, wird das Programm wieder auf-

geweckt, reagiert kurz auf das Ereignis und legt sich wieder »schlafen«, um auf das nächste zu warten.

»Beams« läuft unterdessen mit niedriger Priorität, was bedeutet, daß es rechnet, solange andere Programme im Wartezustand sind, und selber warten muß, wenn mal ein anderes Programm etwas zu erledigen hat. Sie können also normal weiterarbeiten, während »Beams« im Hintergrund läuft und nur die CPU-Zeit »frißt«, die sonst sowieso verschenkt wäre. Ein normaler Texteditor beispielsweise hält »Beams« kaum auf, da der wirklich kaum etwas zu tun hat, außer auf Tastenanschläge des Benutzers zu warten. Ein Compiler ist da schon etwas anderes. Der muß bei längeren Programmen selbst viel rechnen. In der Zeit wird »Beams« kaum weiterkommen, aber - auf die Gesamtrechenzeit für ein Bild umgerechnet - fallen die paar Minuten, die »Beams« vom Compiler aufgehalten wird, kaum ins Gewicht.

Es ist selbstverständlich idealisiert, wenn ich behaupte, daß »Beams« die übrigen Anwendungen des Amiga kaum behindert. Das mag auf die CPU-Benutzung zutreffen; aber »Beams« verbraucht auch Speicher, und den nicht zu knapp. Aus Sicht des Speicherverbrauchs kann »Beams« Ihren Amiga erheblich blockieren, vor allem, wenn man nur wenig Speicher hat. Aber irgendwo ist man wohl immer eingeschränkt. Der Speicherverbrauch hängt sehr stark von dem Umfang, aber auch von der Art der Szene ab, die man »raytracen« will. Wenn Sie also nicht zu reichlich mit Speicher gesegnet sind und noch während des Raytracings etwas anderes machen wollen, würde ich Ihnen raten, nicht zu umfangreiche Szenen berechnen zu lassen bzw. es nur dann zu tun, wenn der Amiga frei ist.

4.2 Vorverarbeitung durch Grid

Die Szene, die von »Construct« erzeugt wurde, ist in einer Form gespeichert, die für das Raytracing-Verfahren nicht geeignet ist. »Beams« braucht zum Arbeiten eine Datenstruktur, die ihm - grob gesagt - mitteilt, wo im Raum welches Objekt liegt. Die Datenstruktur braucht »Beams«, um schnell herauszufinden, welches Objekt welchen Strahl blockiert.

Erzeugt wird die Datenstruktur durch das Programm »Grid«. Das wird automatisch vor »Beams« aufgerufen, wenn die Datenstruktur noch nicht existiert. »Grid« liest die Szene, die in der Datei »szene« gespeichert ist, erzeugt die Datenstruktur und speichert sie unter »szene.vor« ab. Diese Datei kann dann von »Beams« gelesen werden.

Mit »Grid« haben Sie praktisch nichts zu tun. »Grid« braucht als Eingabe nur den Namen der Szenendatei. Seine Parameter, die die Erzeugung der Datenstruktur steuern, gewinnt er aus der Szene (das heißt Anzahl und Art der Objekte, Lage zueinander etc.) sowie dem vorhandenen Speicher auf dem Amiga. Zum Speicher ist jetzt allerdings einiges zu sagen:

Die Datenstruktur, die von »Grid« erzeugt wird, kann recht umfangreich werden. Im Durchschnitt belegt sie zwischen zwei- und fünfmal soviel Speicher wie die eigentliche Szene. Dieser Speicher muß a) im Amiga und b) dort, wo die Szene gespeichert ist, also auf Diskette oder Festplatte, vorhanden sein. Ist im Fall b) der Speicher nicht vorhanden, ist das schlecht, denn dann war die Arbeit von »Grid« umsonst. Im Fall a) ist es nicht ganz

so schlimm. Falls der RAM-Speicher im Amiga für die Datenstruktur nicht mehr ausreicht, fängt »Grid« von vorne an und versucht es auf eine speichergünstigere Weise. Dadurch ist selbstverständlich die Datenstruktur nicht mehr so gut, was sich dann beim Raytracing durch eine längere Rechenzeit bemerkbar macht.

Dummerweise kann »Grid« vorher nicht abschätzen, wieviel Speicher es für die Datenstruktur braucht. Deren Umfang hängt von vielen Faktoren ab. Da ist einmal die Objektanzahl; je mehr Objekte in der Szene sind, desto größer ist die Datenstruktur; das ist einleuchtend. Aber einen größeren Einfluß als die Anzahl hat die Form und die Lage der Objekte zueinander.

So ist beispielsweise die Form der Dreiecke ein wichtiger Faktor. Lange dünne Dreiecke mag »Grid« überhaupt nicht. Die machen viel Arbeit und verbrauchen eventuell auch viel Speicher. Auch wenn sich viele Objekte durchdringen, wird viel Rechenzeit und Speicher benötigt. Das sind aber nur vorsichtige Tips, denn wenn Sie eine Szene designen, kümmern Sie sich ja nicht unbedingt um die Form der Dreiecke, oder darum, ob sich Objekte durchdringen. Sie haben vielmehr den Eindruck der Gesamtszene im Sinn. Nur bei großen Objektanzahlen kann eine Häufung von »ungünstigen« Objekten zur Beeinträchtigung von »Grid« bzw. »Beams« führen.

Maximal kann die Datenstruktur von »Grid« eine Größe von 320 Kbyte erreichen. Das ist aber nur ein theoretischer Maximalwert. Die Datenstruktur teilt sich in drei Bereiche von zweimal 128 Kbyte und einmal 64 Kbyte auf. Wenn »Grid« bei einem der Bereiche an die Grenze kommt, bricht er bereits ab und versucht es noch mal mit speichergünstigeren Verfahren. »Grid« bricht aber auch ab, wenn auf dem Amiga zur Laufzeit kein Speicher mehr zu bekommen ist. Er geht dann davon aus, daß Sie dringendere Programme laufen haben, und versucht es noch mal, diesmal mit weniger Speicher.

Da »Grid« zu Beginn nicht wissen kann, wieviel Speicher die Datenstruktur verbrauchen wird, holt es sich den Speicher immer erst bei Bedarf in kleinen Portionen. Es wäre also nicht sehr klug, während des Laufes von »Grid« eine andere speicherintensive Anwendung (zum Beispiel »Construct«) zu starten, wenn abzuschätzen ist, daß dann der Speicher knapp wird.

Insofern beeinträchtigt »Grid« Sie selbstverständlich bei Benutzung des Amiga, aber die Laufzeit von »Grid« ist, verglichen mit »Beams«, immer noch recht kurz. Als grober Richtwert läßt sich ca. eine Minute Rechenzeit pro 100 Objekte angeben. Das ist aber nur ein ungefährer Erfahrungswert, der bei günstigen Verhältnissen unterschritten und umgekehrt bei ungünstigen Szenen auch überschritten werden kann.

4.3 Raytrace-Parameter

»Beams« kann durch viele Parameter gesteuert werden, die Größe und Qualität des Bildes bestimmen (und die Rechenzeit).

Im »Manager«-Fenster erscheinen bei Anwahl des »Beams«-Menüs folgende Gadgets, mit denen die Parameter eingestellt werden können:

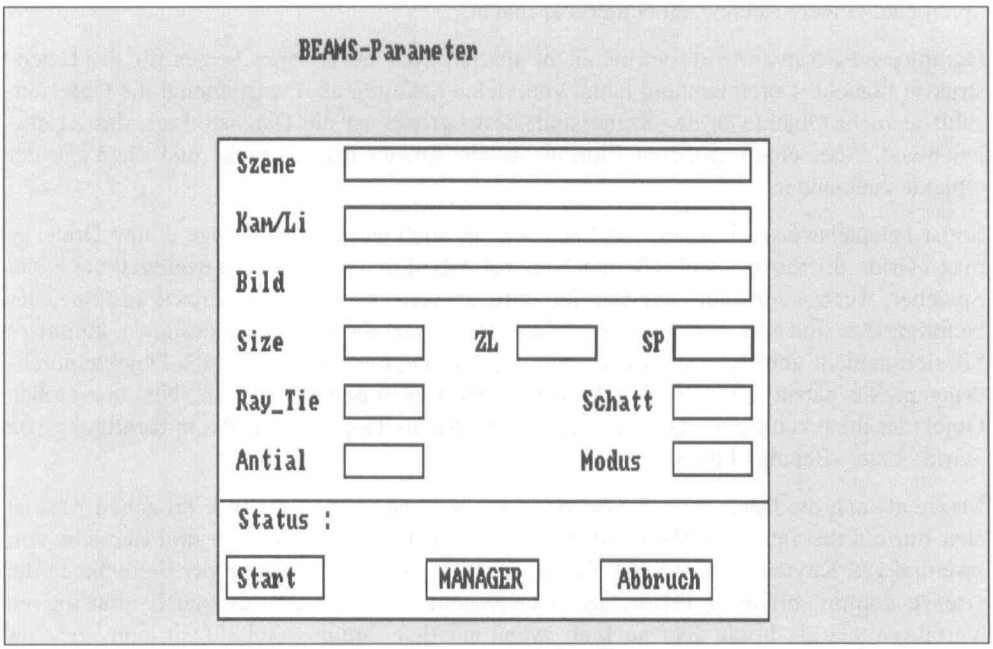


Bild 4.1: Beams-Parameter

Schauen wir uns die einzelnen Parameter genauer an.

Szene: Wenn Sie das Gadget anklicken, können Sie hier den Namen der Szene eintippen, die von »Beams« gelesen werden soll. Hier gehört der Dateiname hin, den Sie bei »Construct« unter »SaveSz« gespeichert haben. Wenn Sie nur den Dateinamen eingeben, sucht »Beams« im Ordner »Reflections/szenen/«. Ist die Szenendatei woanders gespeichert, müssen Sie den vollen Pfad mit angeben (zum Beispiel: »test:objekte/baum«).

Kam/Li: Hier muß der Name der Kamera-Licht-Datei eingegeben werden. Falls die Datei in »Reflections/szenen« ist, genügt der einfache Dateiname, sonst der volle Pfad. (Den Zusatz ».kam« müssen Sie nicht mit angeben, der wird gegebenenfalls automatisch angefügt.)

Bild: In dieses Gadget geben Sie den Dateinamen ein, unter dem »Beams« das erzeugte Bild speichern soll. Falls Sie hier nur den einfachen Dateinamen eingeben, speichert »Beams« das Bild im Directory »Reflections/ bild/«. Wollen Sie es woanders speichern, muß der volle Pfad angegeben werden.

Wenn Sie nebenbei noch anderweitig arbeiten wollen, genug Speicher haben, aber nicht allzu viele Diskettenlaufwerke, dann würde ich die Bilddatei in »ram:« speichern, denn sonst verlangt »Beams« nach jeder gerechneten Bildzeile die Diskette, in der der Ordner »Reflections/bild/« liegt, um das Bild zwischenzuspeichern.

Size: *ZL:* *SP:* Diese drei Gadgets gehören zusammen. Mit ihnen können Sie die Größe des errechneten Bildes festlegen. Durch Anklicken des »Size«-Gadgets können Sie zyklisch eine Anzahl von vordefinierten Bildgrößen wählen. »Beams« kennt folgende:

test1:	32*26	Pixel	(32/4096)	
test2:	64*52	Pixel	(32/4096)	
mittel:	128*104	Pixel	(32/4096)	
LoRes1:	320*256	Pixel	(32/4096)	
LoRes2:	352*290	Pixel	(32/4096)	(Overscan)
HiRes1:	640*256	Pixel	(16)	
HiRes2:	704*580	Pixel	(16)	(Overscan)

In Klammern stehen hinter den Bildgrößen die Anzahl der maximal möglichen Farben, wenn Sie später mit »Show« das Bild in eines der Amiga-Formate umwandeln. Bei Anklicken von »Size« /erscheinen in »ZL« und »SP« jeweils die Anzahl der Zeilen bzw. Spalten bei der Auflösung.

Sie können aber auch eine individuelle Auflösung wählen. Dazu klicken Sie lediglich »ZL« oder »SP« an und tippen dort die gewünschte Zahl ein. Im »Size«-Gadget erscheint darauf der Begriff »Custom«, um anzudeuten, daß jetzt eine benutzerdefinierte Auflösung gewählt wurde.

Ray_Tie: Hiermit wird die Raytrace-Tiefe eingestellt. Klicken Sie das Gadget an und tippen dann die gewünschte Tiefe ein. Unten folgt noch eine nähere Erläuterung zur Raytrace-Tiefe.

Schatt: Dieses Gadget dient zur Ein/Ausschaltung der Schlagschattenberechnung (s.u.). Durch Anklicken wird zwischen »Ein« und »Aus« hin- und hergeschaltet.

Antial: Die Antialias-Berechnung (s.u.) kann hiermit beeinflußt werden. Nach Anklicken des Gadgets müssen Sie den Grad der Berechnung eintippen. »0« schaltet die Berechnung aus. Folgende Antialias-Stufen können Sie angeben:

0: keine Antialias-Berechnung.
1: Antialias nur, wenn »Beams« auf eine Textur stößt.
2: Antialias nur, wenn ein Farbsprung stattfindet und (!) an der Stelle keine Textur ist.
3: Antialias bei Farbsprung.
4: Antialias bei Farbsprung oder (!) bei einer Textur.

Modus: Mit diesem Gadget können Sie die Geschwindigkeit von »Beams« beschleunigen. Die Beschleunigung wird aber nur dadurch erreicht, daß »Beams« schlicht und einfach Pixel ausläßt. Wenn Sie zum Beispiel Modus 10 einstellen, wird nur jedes zehnte Pixel berechnet. Die Pixel dazwischen werden nur mit der Farbe aufgefüllt. Lediglich wenn ein Farbsprung auftritt, werden auch die Pixel dazwischen einzeln berechnet. Modus 0 oder 1 bedeutet normale Berechnung. »Modus« sollte nur bei Szenen mit großen Objekten, bei denen sich farblich nicht viel ändert, angewendet werden, weil sonst mit Sicherheit Fehler im Bild erscheinen.

Ganz unten befinden sich dann noch folgende Gadgets:

Start: Durch Anklicken lösen Sie den Start von »Beams« aus. »Manager« prüft zuvor, ob für die Szene eine Vorverarbeitungsdatei existiert (von »Grid« erzeugt). Wenn nicht, muß »Beams« warten und zuvor wird »Grid« aufgerufen.

Manager: Damit können Sie wieder ins Hauptmenü von »Manager« und dort andere Programme aufrufen (wenn der Speicher reicht). Dies geht aber nur, wenn »Beams« entweder ruht oder rechnet.

Abbruch: Hiermit können Sie die Bildberechnung abbrechen. Dies ist eine Konzession an die langen Rechenzeiten. Es kommt mitunter vor, daß man seinen Amiga auch für andere Dinge braucht. Das bis dahin berechnete Bild ist aber nicht verloren. »Beams« speichert das Bild unter dem Bildnamen und erzeugt zusätzlich eine Datei, in dem alle Angaben stehen, die zur Fortsetzung der Bildberechnung nötig sind. Diese Datei wird unter dem Namen »bild.abbr« gespeichert (»bild« ist der Name der Bilddatei). Sie können die Bildberechnung fortsetzen, indem Sie »Beams« wieder mit den gleichen Angaben starten. Er prüft zuvor, ob eine Datei »bild.abbr« existiert, holt in dem Fall die entsprechenden Daten und setzt die Berechnung an der Stelle fort, an der Sie unterbrochen haben.
Ein Abbruch-Befehl erreicht »Beams« übrigens nur nach Beendigung einer Bildzeile. Sie müssen also bei einem umfangreichen Bild eventuell einige Minuten warten, bis »Beams« auch tatsächlich abbricht. Falls Sie aber abbrechen und nicht beabsichtigen, das Bild später weiterberechnen zu lassen, dann sollten Sie die Datei »bild.abbr« löschen.

Oberhalb der letzten drei Gadgets steht eine Statusangabe zum augenblicklichen Zustand von »Beams«. Er kennt folgende Zustände:

ruht	»Beams« ist nicht aufgerufen.
gestartet	»Manager« hat den Start von »Beams« ausgelöst.
Grid	»Beams« wartet auf »Grid«, der die Vorverarbeitungsdatei erzeugt.
Grid2	»Grid« ist der Speicher ausgegangen. Er versucht es noch einmal mit weniger Speicher. »Beams« wartet weiterhin.
check	»Beams« macht einige Vorberechnungen, reserviert Speicher, lädt Szenen-, Kamera/Licht- und Texturdateien sowie die Vorverarbeitungsdatei.
zeile..	»Beams« arbeitet und sendet an »Manager« immer die Zeile, die gerade in Bearbeitung ist.

Außerdem wird hier angezeigt, wenn Fehler auftreten:

kein Ram	Es ist nicht genügend Speicher vorhanden, um Szene, Vorverarbeitungs-Datei und Textur-Dateien zu laden. (Löschen Sie alle anderen Anwendungen aus dem Amiga, verwenden Sie weniger umfangreiche Texturen. Wenn das auch nicht reicht, müssen Sie sich beim Umfang der Szene etwas bescheiden.)
<Datei>??	»Beams« kann die Szenendatei »Datei« nicht finden.
<Datei> mem	Die Szene ist zu umfangreich.
tex <Datei> ??	Die Texturdatei »Datei« ist nicht im Ordner »Reflections/text/« gespeichert oder es ist keine korrekte IFF-Bilddatei.
tex <Datei> mem	Kein Platz mehr für die Textur-Datei.
Grid-Datei ??	Die Vorverarbeitungsdatei wurde nicht gefunden.
Grid-Datei mem	Kein Platz mehr für die Vorverarbeitungsdatei.
Kam <Datei> ??	Die Kamera/Licht-Datei »Datei« oder »Datei.kam« wurde nicht gefunden.
mat <Datei> ??	Die Materialdatei »Datei« wurde nicht gefunden.
bild <Datei>??	Die Bilddatei (in der das Bild gespeichert wird) konnte nicht geöffnet werden) oder ein Fehler beim Speichern ist aufgetreten.

Falls solch ein Fehler auftritt, bleibt Ihnen nur übrig, das »ok«-Gadget (das normale Start-Gadget) anzuklicken, den Fehler zu beheben und es erneut zu versuchen.

Zu den Raytrace-Parametern sollte ich wohl noch einige Anmerkungen machen:

Die Bildgröße muß wohl nicht näher erläutert werden, außer vielleicht folgendes: Der HAM-Modus läßt sich nur bei einer Bildschirmauflösung von 352*290 oder weniger verwenden. Nur damit ist die volle Benutzung aller 4096 Farben möglich. Bei einer höheren Auflösung sind nur 32 oder 16 Farben möglich (siehe auch Kapitel 5, »Show«).

Ein Parameter, der die globale Bildqualität stark beeinflußt, ist die Raytrace-Tiefe. Sie bestimmt, wie oft ein Strahl an spiegelnden Oberflächen reflektiert bzw. an durchsichtigen gebrochen wird. Man könnte selbstverständlich im Sinne einer naturgetreuen Darstellung fordern, den Strahl prinzipiell zu reflektieren oder zu brechen, wenn er auf ein entsprechendes Objekt trifft. Das entspräche einer Raytrace-Tiefe von unendlich. Es sind eher praktische Erwägungen, wie die Begrenztheit des Speichers und die hohe Rechenzeit, die dazu führen, dem Raytracer eine maximale Raytrace-Tiefe vorzugeben. Vielleicht möchte man auch ganz auf Spiegelung und Brechung verzichten, dann setze man die Raytrace-Tiefe auf 0. Im übrigen ist die Raytrace-Tiefe nicht nur als Begrenzung nach oben hin zu sehen, sondern auch als Schwelle nach unten. Für bestimmte Situationen braucht man nämlich eine minimale Raytrace-Tiefe. Um beispielsweise durch eine Glaskugel durchschauen zu können, ist mindestens eine Tiefe von 2 nötig. Erstens, um mit dem Sehstrahl in die Kugel einzudringen, zweitens, um auf der anderen Seite wieder herauszukommen.

Ein zweiter globaler Parameter bei der Bildberechnung ist die Schlagschattenberechnung. Die kann bei »Beams« ein- bzw. ausgeschaltet werden. Bei Schattenberechnung wird immer, wenn ein Strahl auf ein Objekt trifft, vom Schnittpunkt aus je ein Strahl zu jeder Lichtquelle geschickt, um festzustellen, ob der Schnittpunkt im Schatten eines anderen Objektes liegt. Es empfiehlt sich daher, bei Schattenberechnung mit der Anzahl der Lichtquellen nicht zu übertreiben, da die Strahlanzahl dann extrem ansteigt und damit auch die Rechenzeit. Ohne Schatten ist die Anzahl der Lichtquellen dagegen weniger ausschlaggebend.

Die beiden letzten Parameter beeinflußten die globale Bildqualität, global deshalb, weil hierbei die gesamte Szene einen Einfluß hat bzw. haben kann. Ein weit entferntes Objekt kann sich in einem anderen Objekt spiegeln oder einen Schatten werfen.

Der folgende Parameter betrifft dagegen die lokale Bildqualität, bei der es um die lokalen Gegebenheiten an einer Stelle der Szene geht. Jeder kennt die »Treppenstufen«, die meist an den Kanten von Objekten auftreten. Sie fallen auf, wenn an der Kante ein relativ großer Farbsprung stattfindet. Sie haben ihre Ursache in der geringen Auflösung eines Bildschirms. Die Treppenstufen werden auch »Alias« genannt. Das Verfahren, um die Alias zu unterdrücken, heißt schlicht und einfach »Antialias-Verfahren«. Dieses Verfahren kann bei »Beams« ausgeschaltet oder in verschiedenen Qualitätsstufen benutzt werden. Tritt irgendwo im Bild von einem Pixel zum nächsten ein starker Farbsprung auf, dann vermutet »Beams« an der Stelle eine Objektkante. Er »schaut« sich das betreffende Pixel dann genauer an, indem er ein Bündel von Strahlen durch das eine Pixel schickt. Die Farben, die die Strahlen zurückliefern, werden anteilsmäßig gemischt. Durch die Mischfarbe verwischt die Treppenstufe und die Objektkante erscheint glatt (siehe auch Kapitel 7).

Ein Problem für das Verfahren sind die Texturen. Texturen sind IFF-Bilder, die auf Objekte projeziert werden und die Farbe des Objektes bestimmen. Durch Texturen können auf Objekten beliebig feine Farbmuster entstehen. Dies kann dazu führen, daß ein großes Objekt, das eine feine und farbenreiche Textur besitzt, vollkommen mit dem Antialias bearbeitet werden muß, was die Rechenzeit selbstverständlich gewaltig erhöht. Ein anderer

Fall wäre eine farbenarme Textur, die aber dünne Linien enthält. Da kann es oft vorkommen, daß die Linien übersprungen werden und im Bild nicht oder nur lückenhaft erscheinen. Hier wäre Antialias unbedingt erforderlich.

Da Sie als Benutzer mehr über die verwendeten Texturen wissen als »Beams«, legt er die Entscheidung über die Art des Antialias in Ihre Hände. Sie können in fünf verschiedenen Stufen bestimmen, wann das Verfahren angewendet werden soll.

Antialias:

0: kein Antialias. Sollte man immer bei Testbildern verwenden.

1: Antialias immer, wenn »Beams« mit dem Strahl vom Augenpunkt auf eine Textur stößt. Es sollte angewendet werden, wenn Farbsprünge nicht beachtet werden sollen, aber wenn Feinheiten auf Texturen so gut wie möglich erscheinen sollen.

2: Antialias nur, wenn ein Farbsprung bei benachbarten Pixeln auftritt, aber nur, wenn das betreffende Objekt keine Textur hat. Dies empfiehlt sich, wenn großflächige Texturen vorkommen, bei denen man auf das Antialias aus Rechenzeitgründen verzichtet.

3: Antialias immer bei Farbsprung. Dies ist das »normale« Antialias. Es reagiert auf einen Farbsprung benachbarter Pixel, egal ob eine Textur im Spiel ist.

4: Antialias immer bei Farbsprung und immer bei Auftreten einer Textur. Dies ist die höchste Stufe. Empfehlen kann man die nur, wenn im Bild kleinflächige Texturen vorkommen, die aber möglichst genau erscheinen sollen. Bei einer großflächigen Textur, die eventuell das ganze Bild bedeckt, können Sie sich getrost auf Rechenzeiten von über 10, 12 Stunden und mehr einstellen.

Es ist auch möglich, »Beams« über das CLI aufzurufen (unter Umgehung von »Manager«). Dies bringt vor allem eine Einsparung von Speicherplatz. In dem Fall fragt »Beams« selber nach den Parametern, die Sie dann über Tastatur eingeben müssen. Allerdings müssen Sie sich dann selbst darum kümmern, ob eine Vorverarbeitungsdatei existiert (»szene.vor«). Falls nicht, können Sie auch vom CLI aus »Grid« aufrufen. Wenn »Grid« gestartet ist, will er nur den Namen der Szenendatei wissen.

»Beams« fragt die Parameter in folgender Reihenfolge ab:

- Name der Szenendatei ?
- Name der Kamera/Licht-Datei ?
- Anzahl der Spalten und Zeilen des Bildes
- Schattieren (j=ja,n=nein)
- Raytrace-Tiefe
- Antialias-Stufe (0..4)
- Modus
- Name der Bild-Datei

Sie können die Parameter auch in eine Textdatei ablegen (mit »ED« von der Workbench zum Beispiel) und die Eingabe von »Beams« auf die Datei umlegen. Folgende Textdatei »beams.txt« käme in Frage:

```
Reflections/szenen/testszene
Reflections/szenen/testszene.kam
320 256
j
1
3
0
ram:bild
```

Mit dem Befehl

```
run Ref:Reflections/beams <beams.txt
```

können Sie »Beams« im Hintergrund laufen lassen und mit dem CLI weiterarbeiten. Beim Aufruf durch das CLI behält »Beams« die aktuelle Priorität. Wenn Sie mit dem CLI weiterarbeiten wollen, sollten Sie ihre Priorität mit dem Befehl »ChangeTaskPri« danach etwas höher setzen. Wird »Beams« vom »Manager« aus aufgerufen, dann bekommt er, sobald alle Parameter gelesen sind, automatisch die Priorität -2, so daß das Weiterarbeiten mit dem Amiga nicht mehr behindert wird.

4.4 Speicherverbrauch und Rechenzeiten

Der Speicherverbrauch ist bei dreidimensionaler Computer-Grafik immer ein Problem. Bei umfangreichen Szenen sind eben viele geometrische Daten zu speichern. Folgende Werte sollten Sie berücksichtigen, um abzuschätzen, ob eine umfangreiche Szene noch in Ihrem Amiga rechenbar ist:

- »Beams« braucht ca. 70 Kbyte an Programmcode und fest vereinbartem Speicher. Der übrige Speicherverbrauch hängt vom Unfang der Szene ab.
- Pro Punkt werden 24 Byte Speicher reserviert.
- Pro Objekt (Dreieck oder Kugel) sind 48 Byte nötig.
- Jede Textur-Datei (IFF-Bild) wird voll expandiert. Ein Textur-Bild braucht demnach (zeilen*spalten*Anzahl_Bitplanes)/8 Byte an Speicher. Ein Textur-Bild mit einer Auflösung von 320*256 und 8 Farben (das heißt drei Bitplanes) verbraucht also ca. 30 Kbyte. Benutzen verschiedene Texturen das gleiche IFF-Bild, so wird es aber nur einmal in den Speicher geladen. Trotzdem ist es selbstverständlich anzuraten, nicht zu viele verschiedene Voll-Bilder als Texturen zu benutzen (lieber viele kleine Brushes).
- Die Vorverarbeitungs-Datenstruktur (von »Grid« erzeugt) verbraucht zwischen zwei- und fünfmal soviel Speicher wie die eigentlichen Objekt-Daten (Punkte+Objekte).

Rechenzeiten

Falls Sie schon Kapitel 7 gelesen haben, wissen Sie sicherlich bereits, daß Rechenzeiten ohne das dazugehörige Bild nicht viel Aussagekraft haben. So kann ich hier nur allgemeine Aussagen zur Rechenzeit machen. Beim Titelbild des Buches kann ich jedoch konkret werden. Das Bild wurde auf einem Amiga 500 mit normaler Taktfrequenz in etwa sechs Stunden erzeugt. Ansonsten kann ich folgende Richtwerte geben:

- Ein Bild mit doppelter Auflösung (Zeilen und Spalten) braucht viermal soviel Rechenzeit.
- Bei Verwendung von Antialias verdoppelt sich die Rechenzeit gegenüber der Rechenzeit ohne Antialias im Normalfall (nicht zu umfangreiche Texturen).
- Bei Szenen, bei denen nur große Objekte (im Bild) vorkommen, bei denen sich farblich nicht viel ändert, kann durch Setzen des Modus-Parameters sehr viel Zeit gespart werden. Einfache Bilder (zum Beispiel ein paar Kugeln über einer einfarbigen Fläche) können innerhalb einer halben Stunde berechnet werden.
- Viel Hintergrund spart selbstverständlich Rechenzeit, da dann weniger Strahl-Objekt-Schnittests ausgeführt werden.

FARBTEIL

Farbbild 1: Ein aus Dreiecken zusammengesetztes Ei vor und nach der Rundung.

Farbbild 2: Der rechte Würfel wurde zweimal zerknittert.

Farbbild 3: Kugeln mit verschiedenen Materialeigenschaften: metallisch, matt mit Glanzlicht, matt ohne Glanzlicht.

Farbbild 4: Ein Texturmuster

Farbbild 5: Das Texturmuster aus Farbbild 4 wurde auf einen Körper projeziert.

Farbbild 6: Das Texturmuster wurde um eine Kugel gewickelt.

Farbbild 7: Hier wurde das Texturmuster auf einen Zylinder »geklebt«.

Farbbild 8: Ein Beispiel für eine Etikett-Textur.

Farbbild 9: Eine Kugel mit normaler und eine Kugel mit Genlock-Textur.

Farbbild 10: Ein Beispiel für Material-Textur.

Farbbild 11: Weiche Farbübergänge durch Punkt-Textur.

Farbbild 12: Realistische Szenen lassen sich mit Reflections erzeugen.

Farbbild 13: Eine Traumwelt – faszinierend real.

Farbbild 14: Das Weltall – Reflections war hier Schöpfer.

Farbbild 15 + 16: Selbst aufwendige Szenen errechnet Reflections in annehmbarer Zeit. Das Ergebnis ist verblüffend, oder?

Show und Get_IFF

5.1 Einleitung

Der Amiga ist in bezug auf seine Grafikfähigkeiten in seiner Preisklasse ungeschlagen und auch vielen teuren PCs überlegen. Er kann bis zu 4096 Farben auf dem Bildschirm gleichzeitig darstellen.

Eine hohe Farbtiefe ist die Voraussetzung für realistisch wirkende Bilder. Allerdings sind 4096 Farben, verglichen mit dem, was das menschliche Auge aufnehmen kann, noch recht wenig. In professionellen Systemen wird beispielweise mit 16 Millionen Farben gearbeitet. Das sind je 256 Abstufungen für Rot, Grün und Blau. Das liegt in etwa in der Größenordnung dessen, was das Auge unterscheiden kann. Eine so hohe Farbtiefe verbraucht selbstverständlich auch viel Speicher. Für 16 Millionen Farben braucht man 24 Bitplanes. Der Amiga schafft gerade 6.

»Beams« rechnet übrigens intern mit 24 Bitplanes. Die Bilddatei, die von »Beams« erzeugt wird, enthält noch die volle Farbtiefe und kann daher auch recht umfangreich werden. Die Aufgabe von »Show« ist es nun, das Bild für die verschiedenen Darstellungsarten des Amiga umzurechnen. Es versucht, die Farben so gut wie möglich an die Bildfarben anzupassen bzw. so gut wie möglich zu mischen. Dabei geht selbstverständlich Farbinformation verloren, aber das läßt sich nicht vermeiden.

Befassen wir uns zuerst mit den Möglichkeiten, die der Amiga anbietet.

5.2 Darstellungsarten beim Amiga

Der Amiga kennt viele Darstellungsarten, die sich bei der Bildschirmauflösung und der Farbanzahl unterscheiden. Beide sind allerdings miteinander gekoppelt. Eine höhere Auflösung beschränkt die Farbanzahl.

Der Amiga kennt horizontal und vertikal je zwei Auflösungen. Horizontal kann man normalerweise zwischen 320 (LoRes) oder 640 Pixeln (HiRes) wählen. LoRes steht für Low-Resolution (niedrige Auflösung) und HiRes dementsprechend für High-Resolution.

Vertikal sind 256 oder 512 Pixel möglich. Bei Benutzung dieser Standard-Auflösungen bleibt am Bildschirmrand immer ein Rahmen übrig. Der kann auch noch übermalt werden. In dem Fall sind horizontal 352 Pixel bei LoRes und 704 Pixel bei HiRes möglich. Vertikal ergeben sich dann 290 oder 580 Pixel.

Die Auflösung des von »Beams« berechneten Bildes wird ja bereits beim Start von »Beams« festgelegt. Dadurch ist »Show« schon festgelegt bezüglich der Anzahl der maximal möglichen Farben. Deswegen können Sie sich für »Beams« folgende Regeln merken:

- Bei einer horizontalen Auflösung größer als 352 Pixel sind maximal 16 Farben möglich.
- Der HAM-Modus (4096) Farben ist nur bei Bildern erlaubt, die horizontal höchstens 352 und vertikal höchstens 290 Pixel haben.
- Wird auf HAM verzichtet, sind bei horizontaler Auflösung von 352 Pixel oder weniger maximal 32 Farben möglich.

Der HAM-Modus ist für uns Raytracer wegen seiner Farbvielfalt sicher am interessantesten. Er ermöglicht 4096 Farben auf dem Bildschirm. Allerdings ist die Sache nicht ganz einfach, denn die Farbanzahl wurde beim Amiga nur durch einen Trick erreicht. Normalerweise bräuchte man für 4096 Farben 12 Bitplanes. Der Amiga schafft es auch mit 6. HAM ist eine Abkürzung für »Hold And Modify«, also halte und verändere. Gehalten werden dabei zwei Komponenten der Farbe und verändert wird die dritte. Welche dabei verändert wird, ist frei wählbar. Man kann also nicht von einem zum nächsten Pixel beliebig eine neue aus 4096 Farben wählen. In einem ungünstigen Fall braucht man drei Pixel, um zur gewünschten Farbe zu gelangen. Beim ersten Pixelsprung ändert man zum Beispiel die Grün-Komponente, beim zweiten etwa Blau und beim dritten schließlich noch Rot. Das führt zu den für HAM-Bilder charakteristischen Streifen, die auftreten können, wenn sich Farben sprunghaft verändern, oder auch, wenn das Bild zu viele Farben enthält.

Der Amiga schafft 4096 Farben maximal. Wenn Sie auf HAM verzichten, können die Bilder aus 32 bzw. 16 Farben aufgebaut werden. Diese 32 oder 16 Farben können aus den 4096 Farben ausgewählt werden, die dem Amiga zur Verfügung stehen.

Wieso eigentlich 4096? 4096 entspricht gerade 16*16*16. Und 16 Helligkeitsstufen kann jede der drei Komponenten Rot, Grün oder Blau annehmen. Auf was ich hinauswill: Wenn Sie ein Bild mit nur wenig verschiedenen Farben haben, zum Beispiel ein Bild mit einer großen blauen Kugel, dann entstehen durch die Beleuchtungsfunktion sehr viele verschiedene Blautöne. Von Hellblau, an der dem Licht zugewandten Seite bis zu tiefem Dunkelblau an der abgewandten. Diese Blautöne sollten eigentlich sanft ineinander übergehen, das tun sie aber nicht, da nur 16 verschiedene Blautöne zur Verfügung stehen. All die 4096

Bild 1

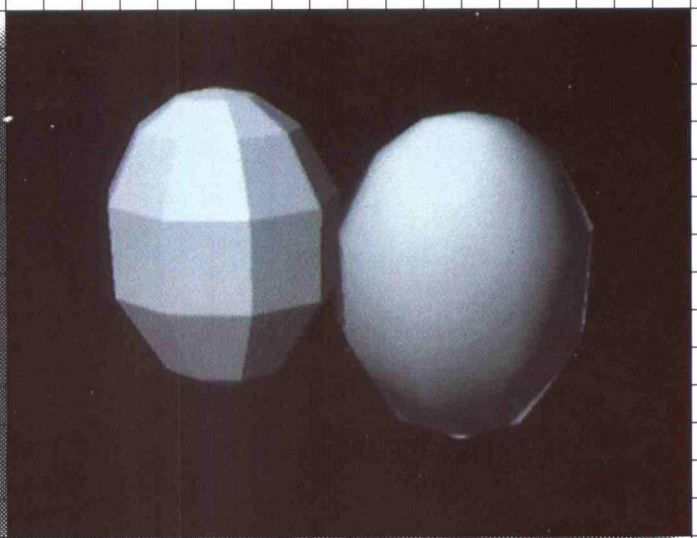


Bild 2

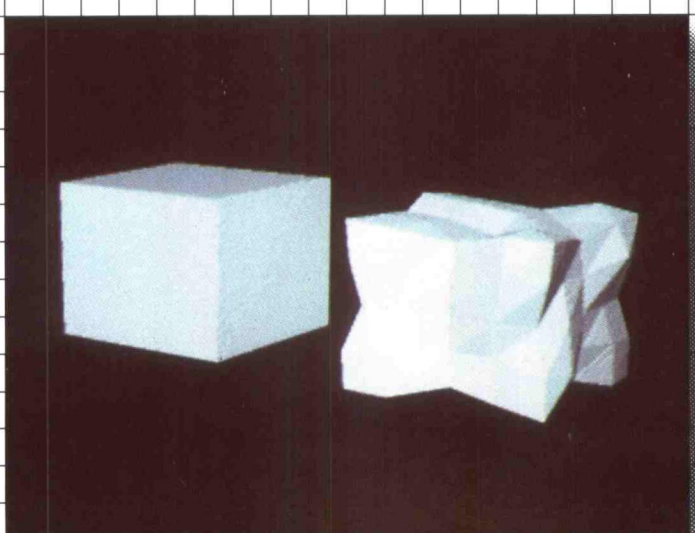


Bild 3

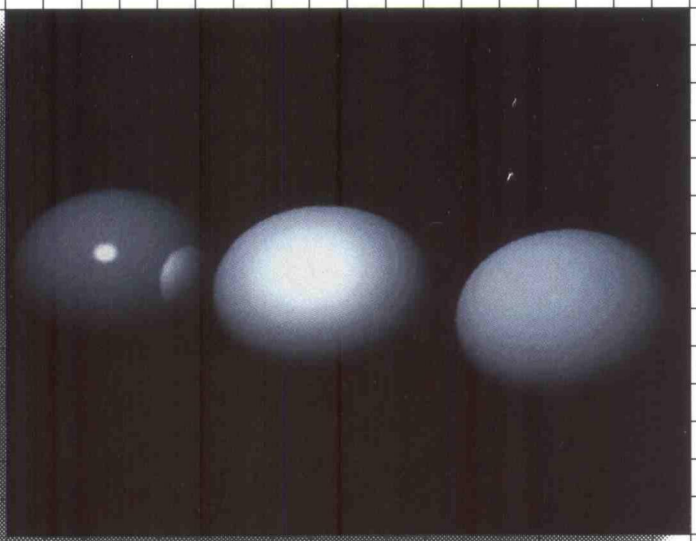


Bild 4

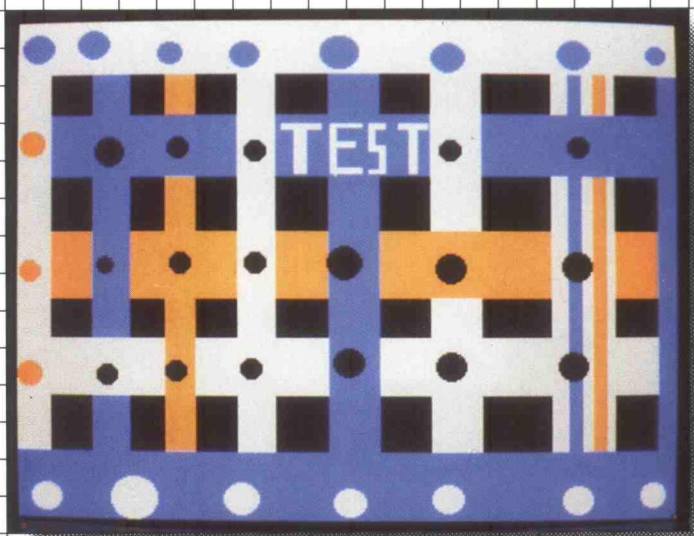


Bild 5

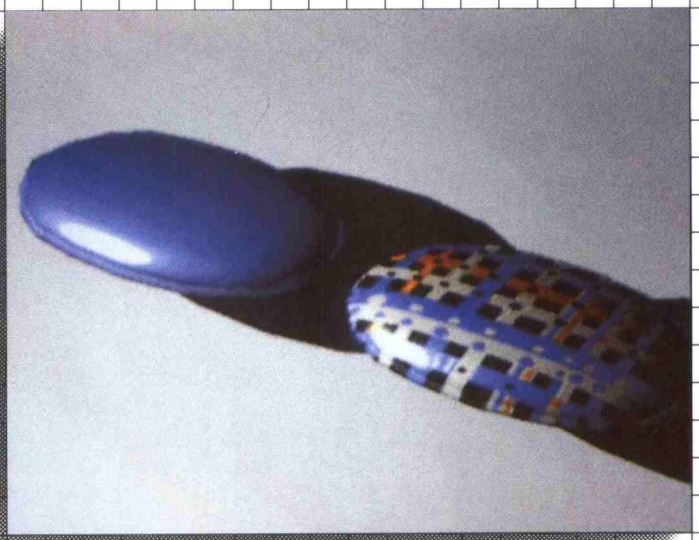


Bild 6

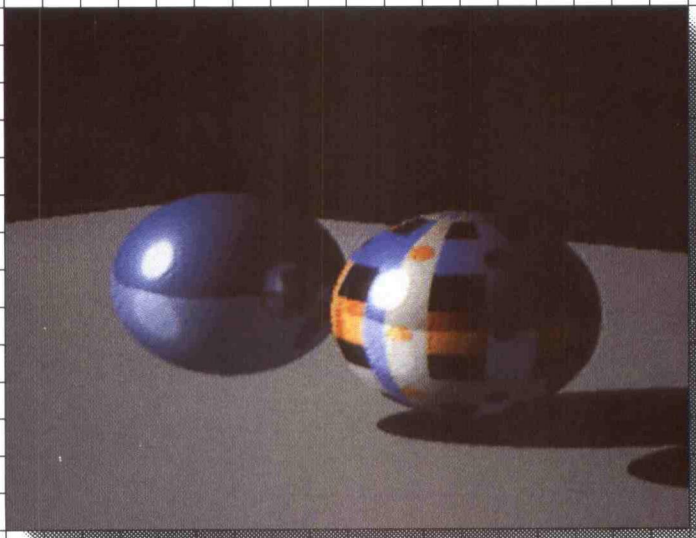


Bild 7

Bild 8

Bild 9

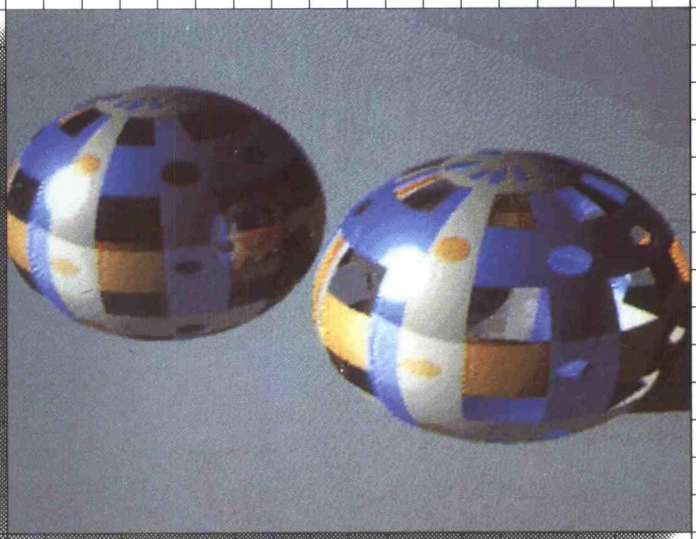


Bild 10

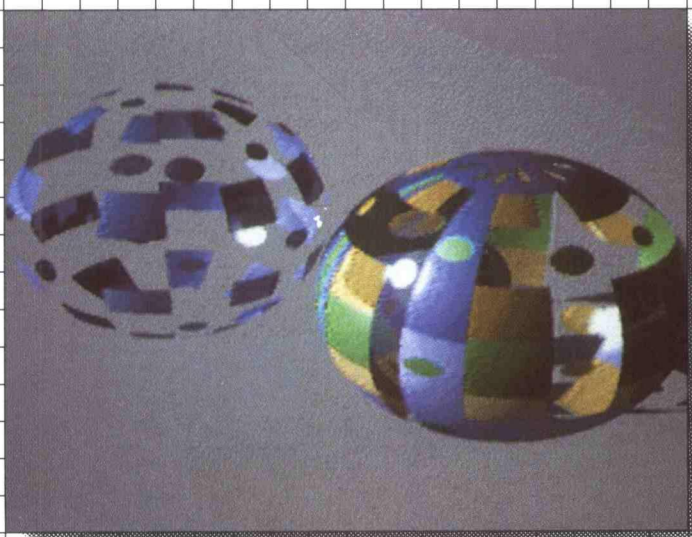


Bild 11

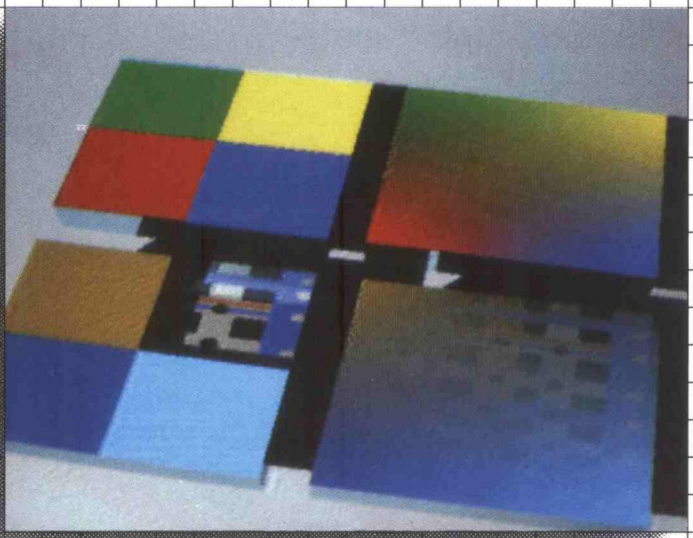


Bild 12

Bild 13

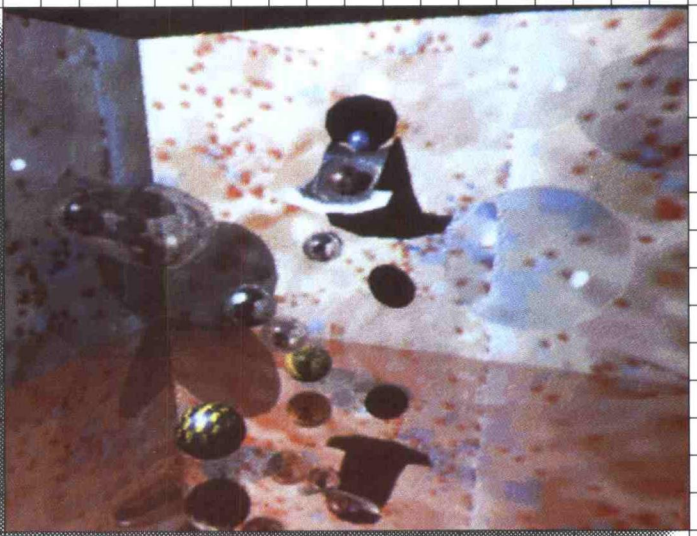


Bild 14

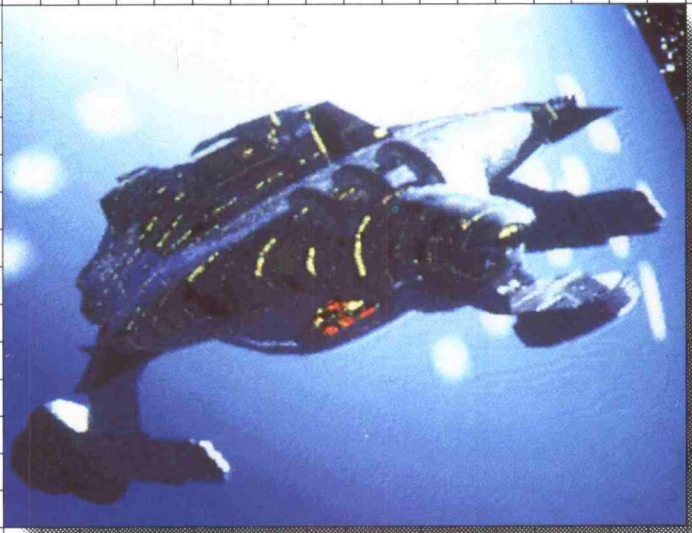


Bild 15

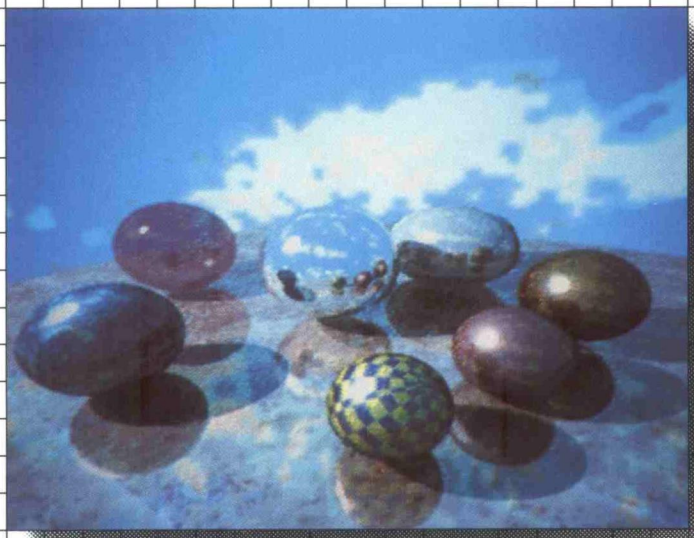


Bild 16

Farben nützen uns herzlich wenig, wenn im Bild nur die Farben einer Komponente vorherrschen. Das Ergebnis sind in dem Fall deutliche Streifen im Bild, mit klar zu erkennenden Helligkeitsstufen.

In dem Fall gibt es noch ein letztes Mittel. Dabei wird ein relativ glatter Farbübergang zwischen zwei Farbstufen durch Mischen der beiden Farben erreicht. Das Mischen darf man sich selbstverständlich nicht wie das Mischen mit Wasserfarbe vorstellen. Es entspricht vielmehr der Technik, die zum Beispiel beim Druck von Schwarzweißbildern in Zeitungen angewendet wird. Dort kennt man auch nur zwei Helligkeitsstufen, nämlich Schwarz und Weiß. Die Grautöne dazwischen werden durch verschiedene Punktdichten von schwarzen Punkten simuliert. Schaut man genau hin, kann man die einzelnen Punkte erkennen. Genauso macht es »Show«, wenn es eine Mischfarbe zwischen zwei Farbstufen simuliert. Kommt zum Beispiel auf dem Bildschirm eine Farbe Lila vor und stehen nur die Farben Rot und Blau zur Verfügung, so füllt »Show« den entsprechenden Bereich mit blauen und roten Pixeln auf, genau in dem Verhältnis, das dem gewünschten Farbton entspräche.

Allerdings sind beim Amiga die Pixel erheblich größer als bei einer Zeitung die einzelnen Druckpunkte. Wird das Mischverfahren angewendet, erkennt man das deutlich, je nachdem, wie viele unterschiedliche Farben zur Mischung zur Verfügung stehen. Die Bilder wirken dann etwas körnig. Und es ist oft Geschmackssache was mehr stört, die deutlichen Farbstufen bei einem ungemischten Bild oder das körnige Aussehen eines gemischten.

Ich kann auch kein Patentrezept angeben, wann man mischen sollte und wann nicht. Beim HAM-Modus kann man oft darauf verzichten, bei Verwendung von nur 32 oder weniger Farben ist es meist unumgänglich, aber es kommt immer auf die individuelle Szene an.

5.3 Show-Parameter

Das war jetzt einiges zum Hintergrund von »Show«, jetzt zur Handhabung:

Vom »Manager« wählen Sie im Hauptmenü den Punkt »Show«. Das Fenster füllt sich dann mit folgenden Feldern:

Bild Hier geben Sie den Dateinamen des Bildes ein, das von »Beams« erzeugt wurde. Liegt die Datei im Ordner »REFLECTIONS/bild/«, dann genügt der Dateiname. Wenn nicht, muß der volle Pfad angegeben werden.

IFF Hier muß der Name der Datei eingegeben werden, unter dem das Bild im Amiga-Format gespeichert werden soll. Das Bild wird im IFF-Format abgelegt, so daß es von anderen Programmen weiterverarbeitet werden kann.

Farben Hier können Sie eingeben, wieviele Farben das Bild haben soll. Die Farbanzahl hängt, wie oben beschrieben, von der Bildauflösung ab. Sie können selbstverständlich auch weniger Farben als die maximal möglichen nehmen. Wenn Sie allerdings HAM verwenden, spielt die Farbanzahl hier keine Rolle.

ViewMode Hier können Sie durch Anklicken zwischen NRM und HAM hin- und herschalten. HAM wird selbstverständlich nur bei einer Bildauflösung von höchstens 352*290 angenommen. Bei NRM können Sie im obigen »Farben«-Feld die Farbanzahl festlegen. Falls Sie vorhaben, das erzeugte Bild mit Malprogrammen weiterzuverwenden, sollten Sie auf jeden Fall eine Version im NRM-Mode speichern, da einige Programme den HAM-Modus nicht beherrschen.

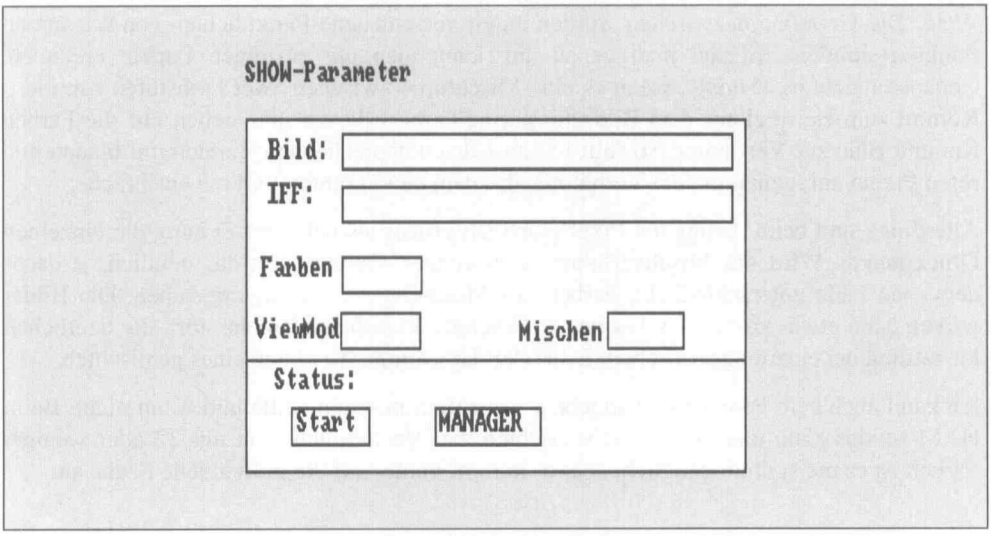


Bild 5.1: Show

Unter dem schwarzen Strich ist das »Start«-Feld. Hier können Sie nach Eingabe aller Parameter »Show« starten. Darüber steht der augenblickliche Status von »Show«. Davon gibt es folgende:

- *ruht* »Show« ist nicht aufgerufen.
- *gestartet* »Manager« hat das Starten von »Show« ausgelöst .
- *check* »Show« prüft die Bild-Datei.
- *rechnet* »Show« ermittelt die optimalen Farben.
- *abspeichern ?* »Show« fragt, ob Sie das Bild unter dem im Feld »IFF« angegebenen Namen speichern wollen. Wenn ja, klicken Sie das »Ja«-Feld an, sonst das »Nein«-Feld.
- *gescheitert* Das Speichern der IFF-Datei ist gescheitert. (Diskette voll oder illegaler Dateiname.)
- *keine Bilddatei* Die angegebene Bilddatei enthält kein korrektes Bild von »Beams«.

Nach Eingabe aller Parameter können Sie das »Start«-Feld anklicken. Darauf liest »Show« die Bilddatei und ermittelt die optimalen Mischfarben. In der Statusanzeige steht dann »rechnet«. Nachdem die Farben ermittelt sind, erscheint ein neuer Bildschirm, der sich langsam mit dem Bild füllt. Sobald das Bild vollständig zu sehen ist, können Sie durch Klicken mit einer der Maustasten den Workbench-Bildschirm wieder in den Vordergrund holen. »Show« fragt dann, ob das Bild als IFF-Datei gespeichert werden soll. Wenn ja, klicken Sie das »ja«-Feld an.

Solange bis Sie ja oder nein anklicken, ist das Bild noch vorhanden, und Sie können es durch Anklicken des Tiefen-Gadgets vom Workbench-Bildschirm jederzeit in den Vordergrund holen.

Falls Sie auf den »Manager« verzichten, können Sie »Show« auch vom CLI aus aufrufen. »Show« fragt dann die Parameter der Reihe nach selbst ab:

- Bilddatei
- Anzahl Farben

Wenn Sie bei der Farbanzahl eine höhere Zahl als 32 angeben, wird – wenn möglich – automatisch der HAM-Modus benutzt. Sobald das Bild fertig ist, fragt »Show«, ob Sie es speichern wollen. Antworten Sie mit [j] für ja oder [n] für nein. Falls ja, müssen Sie noch den Namen der IFF-Datei angeben, unter der das Bild gespeichert werden soll.

5.4 Get_IFF

»Get_IFF« dient nur zum Lesen von IFF-Bildern. Sie können damit die Bilder, die von »Show« erzeugt wurden, anschauen, aber auch andere IFF-Bilder, zum Beispiel die Textur-Muster im Ordner »REFLECTIONS/text/«. Bei Anwahl von Get_iff im »Manager«-Menü füllt sich das Fenster mit den wie in Bild 5.2 dargestellten Feldern.

Hier kann nur der Name der IFF-Bilddatei angegeben werden (im IFF-Feld). Falls Sie nur den reinen Dateinamen angeben, sucht »Get_IFF« automatisch im Ordner »REFLECTIONS/iff/«. Wenn die Datei anderswo liegt, müssen Sie den vollen Dateipfad mit angeben.

Mit dem »Start«-Feld lösen Sie den Ladevorgang aus. Daraufhin wird, falls der Dateiname korrekt ist, das IFF-Bild geladen. Durch Klicken mit der linken Maustaste können Sie den Cursor unsichtbar machen, falls der stört. Wenn Sie mit rechts klicken, kommt der Workbench-Bildschirm wieder nach vorne. »Get_IFF« wartet, bis Sie »Start« anklicken; dann wird das Bild gelöscht.

»Get_IFF« sollte übrigens dafür benutzt werden, die Textur-Muster anzuschauen. Wenn Sie zum Beispiel bei »Construct« ein Textur-Material vergeben wollen und sich nicht über das Aussehen der Textur im klaren sind, kann dem leicht abgeholfen werden:

- Beim »Construct«-Schirm klicken Sie das Tiefen-Gadget an; dadurch kommen Sie wieder zur Workbench. Im Fenster klicken Sie das »Manager«-Gadget an, um ins Menü des »Managers« zu kommen.

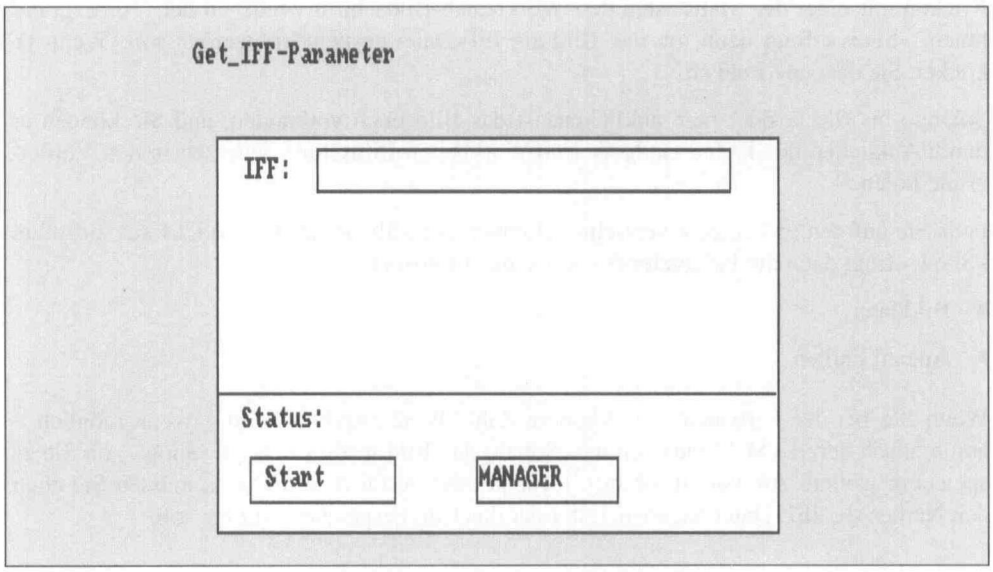


Bild 5.2: Get_IFF-Parameter

- Dort wählen Sie »Get_IFF« und geben im IFF-Gadget den Dateinamen der Textur an. Der Dateiname ist bei der Funktion »EditTex« (bei »Construct«) im Feld »FileName« zu ersehen. Allerdings müssen Sie beim Dateinamen noch den Ordner »REFLECTIONS/text/« vorne anfügen, weil »Get_iff« sonst im Ordner »REFLECTIONS/iff/« sucht.
- Sobald Sie mit dem Bild fertig sind, beenden Sie »Get_iff« wie oben beschrieben.
- Klicken Sie beim Workbench-Bildschirm das Tiefen-Gadget an, um wieder den »Construct«-Screen in den Vordergrund zu holen.

Objekt-Konverter

Reflections ist nicht das einzige Raytrace-Programmsystem. Es gibt bereits einige Programme, die schon einige Zeit auf dem Markt sind. Für die wurden bereits ein Menge von Szenen erzeugt. Um diese Szenen auch in Reflections benutzen zu können, gibt es zwei Hilfsprogramme, um Szenen zweier weitverbreiteter Programme lesen zu können. Die beiden Programme müssen vom CLI aus aufgerufen werden. Sie wandeln die Formate so um, daß sie von »Construct« gelesen werden können.

Get_sculpt

Dieses Programm liest Szenen des Programms »Sculpt-3D«. Der Aufruf von CLI aus sieht folgendermaßen aus:

```
get_sculpt <name>.scene
```

<name> ist der Name einer Szenendatei, die von Sculpt-3D erzeugt wurde. Sculpt-3D hängt an den Dateinamen das Anhängsel ».scene« an. »Get_sculpt« erzeugt eine Datei mit dem Namen »<name>.CONST«, die dann von »Construct« gelesen werden kann.

Bei der Umwandlung verschiedener Formate gibt es aber immer Schwierigkeiten, da unterschiedliche Systeme meist verschiedene Schwerpunkte setzen, die sich selbstverständlich auch im Datenformat niederschlagen. So hat Reflections ein völlig anderes Format bei den Materialeigenschaften, das sich nur sehr umständlich umsetzen ließe. Eine weitere Schwierigkeit findet sich in der Orientierung der Dreiecke. Reflections erwartet die Dreiecksorientierung im Uhrzeigersinn, während die bei Sculpt-3D keine Rolle spielt. In Sculpt-3D gibt es keine Körperverwaltung wie bei Reflections, also kann die Szene nur als ein Körper gespeichert werden. Eventuelle Unterscheidungen müssen dann in »Construct« mittels »Define« vorgenommen werden. Wegen den genannten Einschränkungen liest »Get_sculpt« nur folgende Daten aus dem File:

1) Punkt-Koordinaten.
2) Dreiecksdaten:
   a) Eckpunkte
   b) Rundungsinformation (smooth-Flag)

Alle anderen Werte wie Beobachterdaten (Observer), Weltdaten (World), Kanten (Edge) und Beleuchtungsdaten (Lamp) werden ignoriert und müssen in »Construct« neu gesetzt werden.

Nachdem die Daten eingelesen wurden, muß »Get_sculpt« die Dreiecke, die von Sculpt-3D in willkürlicher Orientierung gespeichert werden, so umdrehen, daß zumindest zusammenhängende Dreiecke gleich orientiert sind. Das Programm kann allerdings nicht wissen, was innen und was außen ist. So kann es vorkommen, daß Dreiecke, die zum Beispiel eine Kugel bilden, danach alle nach innen zeigen, anstatt nach außen. Aber zumindest alle Dreiecke, die zusammenhängen, sind gleich orientiert. Und das ist wichtig für Reflections. In »Construct« können Sie dann die Orientierung der Dreiecke folgendermaßen korrigieren:

- Laden Sie die konvertierte Sculp-Szene. Es existiert nun nur der »all«-Körper.
- Plotten Sie mit »Hidd_2« die Dreiecke. Fassen Sie mit »Define« zusammenhängende Dreiecke unter Körpernamen zusammen.
- Plotten Sie die Körper dann einzeln. Meistens sieht man dann schon, ob sie falsch herum orientiert sind. (Die zugewandten Dreiecke sind gepunktet, die abgewandten durchgezogen gezeichnet.) In dem Fall mit »Umdreh« (im »Körper«-Menü) den Körper umdrehen.

Sie können Kugeln, die in Sculpt-3D aus vielen Dreiecken zusammengesetzt sind, hier löschen und durch echte Kugeln ersetzen.

Put_sculpt

Dieses Programm macht das Umgekehrte wie »Get_sculpt«. Es nimmt eine Szene von »Construct« und wandelt die Objekt-Daten in ein Format um, das von Sculpt-3D gelesen werden kann. Der Aufruf erfolgt folgendermaßen:

```
Put_sculpt <name>
```

»Put_sculpt« wandelt die Daten um und speichert sie unter dem Namen »<name>.scene«. Es werden nur Punkte, Kanten und Dreiecke gespeichert. Kugeln werden ignoriert.

Raytrace-Verfahren

Raytracing ist in letzter Zeit ein häufig gebrauchtes Schlagwort. Es wird meist mit schönen und realistisch wirkenden Bildern in Verbindung gebracht, aber auch mit sehr langen Rechenzeiten. In diesem Kapitel soll das Raytrace-Verfahren näher beleuchtet werden. Es werden die Ursachen für den hohen Rechenaufwand erklärt und Beschleunigungsmethoden aufgeführt. Da man vor allem »raytraced«, wenn man möglichst realistische Bilder erzeugen will, wird auch noch auf Beleuchtungsverfahren eingegangen.

Für die Arbeit mit dem Reflections-System ist dieses und das nächste Kapitel nicht unbedingt notwendig. Wer also gleich loslegen will, kann sie überschlagen und sich die Kapitel 7 und 8 später vornehmen. Es wird aber ein Grundwissen geliefert, das dann sehr nützlich ist, wenn man alle Möglichkeiten des Systems ausnützen will.

Noch ein paar Worte zur Schreibweise in diesem Kapitel:

Es ist selbstverständlich immer ein Problem, zu entscheiden, wie tief man in einen bestimmten Sachverhalt einsteigen soll. Um das Thema erschöpfend auszuführen, müßte man viele mathematische Formeln behandeln, die nur den interessieren würden, der tatsächlich ein Raytrace-Programm schreiben will. Andererseits kann man auch ins Gegenteil verfallen und den Sachverhalt nur mit allgemeinen Stichworten beschreiben, ohne konkret zu werden. Ich werde in diesem Kapitel völlig auf mathematische Schreibweisen verzichten und auch Programmfragmente, die zur Verdeutlichung angeführt werden, nur in Umgangssprache formulieren. Ich denke aber, daß der Leser, der ohnehin tief in der Materie steckt, sich aus der Beschreibung des Verfahrens die Einzelheiten ergänzen kann. Am Ende dieses Kapitels kommt auch noch ein Abschnitt mit Anmerkungen, um da auszuhelfen, wo ein Fachbegriff nicht ganz geläufig ist. Im Text ist dann ein Verweis auf den Abschnitt.

Auf dem Amiga wird noch nicht lange »geraytraced«. Seit 1987 sind die ersten Raytrace-Programme zu haben. Das Verfahren ist jedoch etwa sieben Jahre älter. Es wurde 1980 von Whitted vorgestellt, aber seitdem hauptsächlich auf Großrechnern eingesetzt, eben wegen

des hohen Rechenaufwands. Erst in jüngster Zeit wurden Beschleunigungstechniken entwickelt, die es ermöglichen, auch auf Kleinrechnern dieses Verfahren einzusetzen.

7.1 Der Grundalgorithmus

Der Raytrace-Algorithmus zeichnet sich gegenüber anderen Verfahren durch seine Einfachheit aus. Das Prinzip des Raytracens ist schon im Namen enthalten (Ray = Strahl, to trace = verfolgen, beobachten). Man verfolgt simulierte Lichtstrahlen und schaut, was ihnen auf ihrem Weg passiert. Dabei hält man sich an die physikalischen Gesetze der Strahlenoptik. In der wirklichen Welt gehen von jeder Lichtquelle unendlich viele Strahlen in jede Richtung los. Viele davon treffen auf Oberflächen von Objekten, werden dort gespiegelt oder gebrochen und setzen ihren Weg in eine neue Richtung fort. Einige Strahlen erreichen, nachdem sie von Objekten reflektiert wurden (evtl. mehrmals), das Auge des Betrachters und erzeugen dort den Farbeindruck. Die meisten Strahlen jedoch kommen nie in die Nähe des Betrachters.

Um dies im Rechner zu simulieren, wird ausgenutzt, daß der Strahlenweg umkehrbar ist. Man verfolgt also nur die Strahlen, die vom Auge ausgehen, bis sie auf Lichtquellen stoßen. Das sind allerdings immer noch unendlich viele. Um diese Komplexität praktikabel zu machen, wählen wir aus dieser unendlichen Strahlenmenge die Strahlen aus, die vom Auge durch die Pixel eines gedachten Bildschirms gehen. Das sind beispielsweise bei einer Bildschirmauflösung von 320*256 ca. 82000 Strahlen.

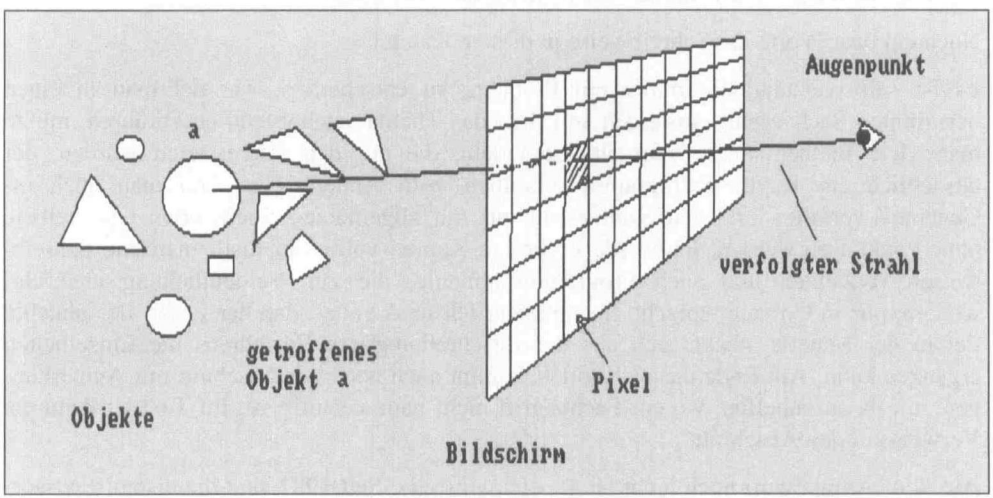


Bild 7.1: Der Grundalgorithmus

Für jeden dieser Strahlen stellen wir nun fest, auf welches Objekt er zuerst trifft. Haben wir das Objekt, können wir aus seiner Farbe, seiner Oberflächenbeschaffenheit und dem Winkel und der Entfernung zur Lichtquelle die Farbe des Pixels bestimmen. Bild 7.1 verdeutlicht das Grundprinzip.

Um zu Anfang eine Grundversion des Raytrace-Verfahrens zu formulieren, schaffen wir uns zunächst einige Probleme vom Hals, oder besser gesagt, wir verschieben sie auf später:

Auf welches Objekt ein Strahl trifft, stellt ein Unterprogramm

Suche_Objekt(s,objnr,x)

fest. Es bekommt als Eingabe einen Strahl »s«, stellt dann fest, auf welches Objekt der Strahl trifft, und liefert in »objnr« die Nummer des Objektes und in »x« die Koordinaten des Schnittpunktes. Wenn der Strahl auf kein Objekt trifft, was oft der Fall ist, dann wird in »objnr« »0« als Objektnummer geliefert.

Die Farbberechnung überlassen wir ebenfalls einem Unterprogramm:

Berechne_Farbe(s,objnr,x,licht,farb)

Es bekommt in »s« einen Strahl, in »x« den Schnittpunkt von »s« mit Objekt »objnr« und in »licht« die Lichtquelle als Eingabe. In »farb« stellt uns das Unterprogramm dann die errechnete Farbe zur Verfügung.

Mit diesen Unterprogrammen kann jetzt eine einfache Version einer Raytrace-Prozedur

Raytrace_0(s,farb)

formuliert werden. Diese bekommt als Eingabe schlicht einen Strahl »s« und liefert in »farb« einen Farbwert. Es wird hierbei eine globale Lichtquelle »l« und ein Hintergrundfarbwert »hintergrund« vorausgesetzt.

```
Raytrace_0(s,farb)
{
    Suche_Objekt(s,Objekt_nr,schnittpkt);
    Falls Objekt_nr>0
         dann Berechne_Farbe(s,Objekt_nr,schnittpkt,l,farb)
         sonst farb=hintergrund;
}
```

Bild 7.2: Raytrace_0(s_farb)

Das ist relativ kurz. Im ersten Schritt stellen wir fest, ob der Strahl »s« ein Objekt trifft, und wenn ja, welches. Falls das zutrifft, wird daraus die Farbe berechnet, falls nicht, ist unser Strahl ins Leere gegangen und wir müssen als Farbe eine Hintergrundfarbe einsetzen. Mit Raytrace_0 kann jetzt auch das Muster eines Bilderzeugungsprogrammes aufgesetzt werden:

werden:

```
Bilderz
{
      für (alle Pixel des Bildschirms)
      {
            s=Strahl vom Augenpunkt zum Pixel
            Raytrace_0(s,f)
            Pixelfarbe(f)
      }
}
```

Bils 7.3: Bilderz

»Bilderz« ist ebenfalls sehr einfach und wird sich auch im folgenden nicht mehr verändern. Wie oben bereits gesagt, erzeugt »Bilderz« für jedes Pixel des Bildschirms einen Strahl, der vom Augenpunkt zu ihm führt. Mit dem Strahl wird jeweils »Raytrace_0« aufgerufen, und die Farbe, die davon zurückgeliefert wird, als Farbe des Pixels im Bildschirm gesetzt. (Einzelheiten, wie das Einlesen der Objekte oder das Speichern des Bildes, sind hier selbstverständlich ausgelassen.)

Mit »Bilderz« könnte man bereits Bilder erzeugen, nur sind Spezialeffekte, wie Schatten, Spiegelung und Transparenz noch nicht berücksichtigt, aber wir sind ja erst am Anfang. Das, was wir mit »Raytrace_0« erzeugen können, läßt sich mit anderen Verfahren (siehe Kapitel 8) ebenfalls berechnen, und zwar sehr viel schneller. Aber mit den Spezialeffekten, die nun eingebaut werden, tun sich die anderen Verfahren viel schwerer.

7.2 Schatten

»Schatten« dürfen Sie nicht mit »schattieren« verwechseln. Der zweite Begriff (im engl. shading) bezeichnet eher die Berechnung der Oberflächenfarbe eines Objektes. Dagegen wollen wir feststellen, ob ein Objekt (genauer gesagt: ein Punkt auf der Oberfläche des Objektes) im Schatten eines anderen Objektes liegt. Wie läßt sich das feststellen? Im Prinzip sehr einfach. Ein Punkt liegt im Schatten eines Objektes, wenn von dem Punkt aus die Lichtquelle nicht mehr zu sehen ist. Einen Strahl, der von dem Punkt in Richtung der Lichtquelle losgeschickt würde, würde das Objekt treffen, in dessen Schatten der Punkt liegt, und damit ist bereits gesagt, wie man die Schattenberechnung in das Raytrace-Verfahren einbauen kann:

Man schickt einfach von jedem Punkt, auf den die Strahlen aus »Raytrace_0« treffen, einen weiteren Strahl zur Lichtquelle. Trifft der Strahl ein Objekt, dann liegt der Punkt im Schatten und erhält kein Licht von der Quelle. Trifft der Strahl nichts, dann wird das Licht, wie zuvor bei »Raytrace_0« mit »Berechne_Farbe« ermittelt.

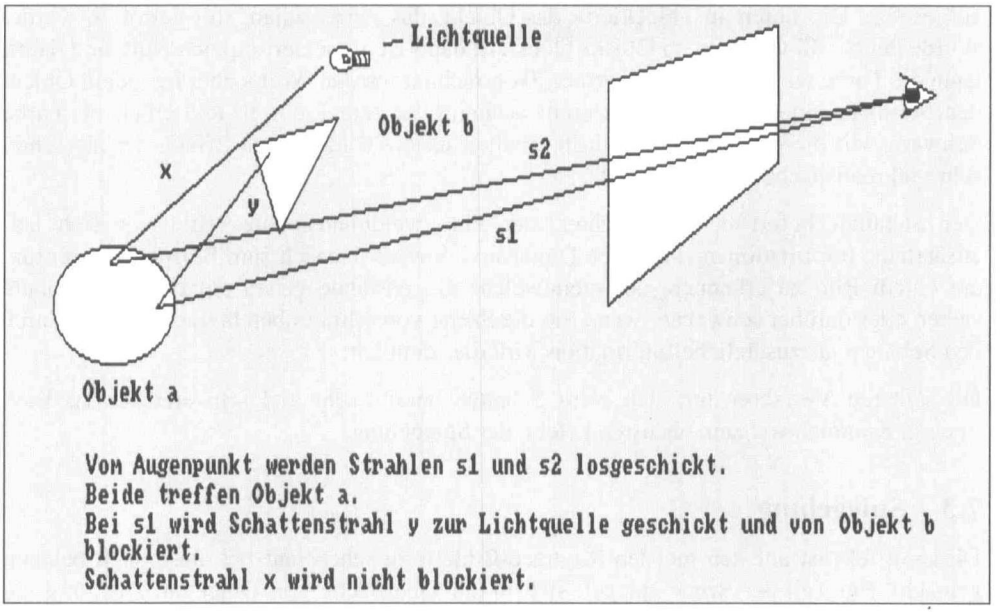


Bild 7.4: Schattenstrahl

Damit erhalten wir »Raytrace_1«:

```
Raytrace_1(s,farb)
{
     Suche_Objekt(s,Objekt_nr,schnittpkt);
     Falls Objekt_nr>0
     dann {
           schatt_str=Strahl von schnittpk zur Lichtquelle
l
           Suche_Objekt(schatt_str,blockiert,schnitt1)
           Falls (blockiert=0)
           dann Berechne_Farbe(s,Objekt_nr,schnittpkt,l,f)
           sonst farb=schwarz
                 }
     sonst farb=hintergrund
}
```

Bild 7.5:Raytrace_1(s,farb)

»Raytrace_1« ist nicht wesentlich komplizierter als »Raytrace_0«. In »schatt_str« wird der

aufgerufen. Die liefert in »blockiert« das Objekt, das vom »schatt_str« getroffen wurde. Wurde der Strahl von keinem Objekt blockiert, dann ist »blockiert« gleich Null, und damit kann die Farbe wie zuvor bei »Raytrace_0« errechnet werden. Wenn aber irgendein Objekt den Strahl blockiert hat, dann bekommt »schnittpkt« kein Licht ab und erhält als Farbe Schwarz. Mit diesem einfachen Schritt erhalten unsere Bilder einen Effekt, der sie schon sehr viel realistischer erscheinen läßt.

Der Schatten liefert dem Betrachter, der ein zweidimensionales Bild vor sich hat, zusätzliche Informationen zur dritten Dimension. Sie werden sich zum Beispiel schwer tun, auf einem Bild zu erkennen, ob irgendwelche Gegenstände genau auf einer Tischplatte stehen oder darüber schweben, wenn Sie die Szene von schräg oben betrachten. Erst durch den Schatten als zusätzliche Information wird dies deutlich.

Die anderen Verfahren tun sich beim Schatten bereits sehr viel schwerer als das Raytracing. Kommen wir zum nächsten Effekt, der Spiegelung.

7.3 Spiegelung

Dieser Effekt ist auf den meisten Raytrace-Bildern zu sehen und hat diese auch bekannt gemacht. Ein Teil der Szene spiegelt sich in der Oberfläche von Objekten. Dies trägt zu einem erheblichen Maß zur Realitätsnähe des Bildes bei. Es ist doch so, daß nahezu alles, was aus Kunststoff, Metall oder Glas hergestellt wird, mehr oder weniger spiegelt. Und meistens werden beim Raytracing Bilder künstlicher Gegenstände erzeugt.

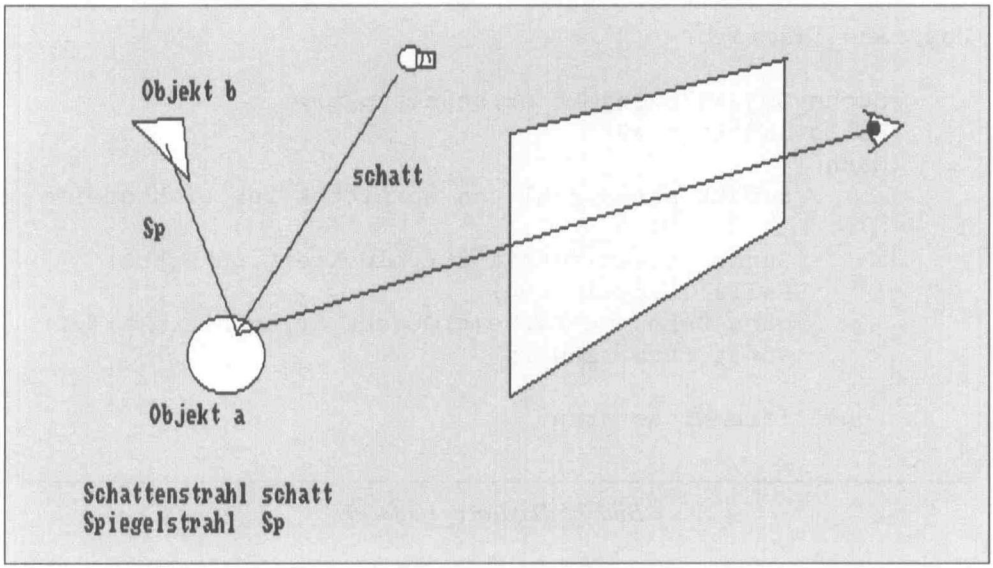


Bild 7.6: Spiegelung

Die Spiegelwirkung kann fast ebenso einfach wie die Schattenberechnung in die Raytrace-Routine eingebaut werden.

Schaut man aus einer bestimmten Richtung in einen Spiegel, so wird der »Sehstrahl« vom Auge zum Spiegel an der Spiegeloberfläche reflektiert, und man sieht, auf was dieser reflektierte Strahl trifft.

Die Richtung, in die der Sehstrahl reflektiert wird, ist nach einem Grundgesetz der Strahlenoptik leicht zu berechnen (7.10.d). Wir fassen die Berechnung dieses Reflektionsvektors wieder in einem Unterprogramm

Reflekt_Strahl(s,objnr,x,reflekt)

zusammen. Sie erhält als Eingabe den Sehstrahl »s«, die Nummer »objnr« des Objektes, auf den er getroffen ist, den Schnittpunkt »x« von »s« mit »Objnr«. In »reflekt« wird dann der Reflektionsstrahl geliefert. Wie das ganze in das Raytrace-Programm eingebaut wird, ahnen Sie wahrscheinlich schon. Man schickt vom Schnittpunkt des Sehstrahls mit der Objektoberfläche wieder einen Strahl los.

```
Raytrace_2(s,farb)
{
     Suche_Objekt(s,Objekt_nr,schnittpkt)
     Falls Objekt_nr>0
     dann {
          schatt_str=Strahl von schnittpk zur Lichtquelle
l
          Suche_Objekt(schatt_str,blockiert,schnitt1)
          Falls blockiert=0
          dann
Berechne_Farbe(s,Objekt_nr,schnittpkt,l,f1)
          sonst f1=schwarz
          Reflekt_Strahl(s,Objekt_nr,schnittpkt,r_strahl)
          Raytrace_2(r_strahl,f_spieg)
          f=f1+f_spieg
     }
     sonst f=hintergrund
}
```

Bild 7.7: Raytrace_2(s,farb)

Hinzugekommen ist hier folgendes: Nach der Berechnung des Schattens wird mit »Reflekt_Strahl« die Reflektion des Sehstrahls »s« auf der Oberfläche von Objekt »nr« errechnet und in »r_strahl« gespeichert. Danach ruft sich »Raytrace_2« selbst auf, diesmal

mit dem »r_strahl« als Eingabe. Damit tut »Raytrace_2« so, als wäre es selbst der Beobachter, und möchte wissen, wie es in Richtung »r_strahl« aussieht. Die Farbe, die es dabei erhält, wird in »f_spieg« zwischengespeichert und dann mit der Farbe »f1«, die von der Lichtquelle kam, zur endgültigen Farbe »f« zusammengerechnet.

Es wurde zwar nicht viel (vom Unfang her) hinzugefügt, aber weil sich »Raytrace_2« jetzt selbst aufruft, kann man schon von einer strukturellen Änderung sprechen, denn »Raytrace_2« ist damit eine rekursive Prozedur.

Beim »inneren« Aufruf von »Raytrace_2« kann der Strahl erneut auf ein spiegelndes Objekt stoßen und muß dann erneut »Raytrace_2« aufrufen. Es kann somit eine ganze Kette von ineinanderverschachtelten »Raytrace_2«-Aufrufen entstehen.

Sie fragen sich vielleicht, wie denn dieses »sich Aufrufen« von »Raytrace_2« endet. Spätestens, sobald ein Reflektionsstrahl nichts mehr trifft, wird für ihn die Hintergrundfarbe genommen. Dann baut sich die ganze Aufrufkette von hinten nach vorne wieder ab. Aber es entsteht ein Problem, wenn zum Beispiel ein Strahl zwischen zwei parallelen Spiegeln ständig hin und her reflektiert wird. Dann würde sich »Raytrace_2« selbst bis in alle Ewigkeit (oder bis der Speicher voll ist) selbst aufrufen. Hier wird in der Tat noch eine Sicherung eingebaut. »Raytrace_2« zählt mit, wie oft es sich bereits selbst aufgerufen hat, und ab einer bestimmten Anzahl bricht es die Kette ab. Diese Anzahl wird auch Raytrace-Tiefe genannt und ist Ihnen in Kapitel 4 bei der Beschreibung von »Beams« begegnet, wo Sie sie als Parameter von »Beams« setzen können. Dies sei aber nur am Rande eingeflochten. Das war mehr ein Implementierungsdetail, und derartige Einzelheiten übergehen wir großzügig.

7.4 Brechung

Nun fehlt nur noch die Lichtbrechung an durchsichtigen Oberflächen. Die Vorgehensweise ist prinzipiell genauso wie bei der Spiegelung. Nur muß diesmal kein Reflektionsstrahl, sondern ein Brechungsstrahl berechnet werden. Dafür nehmen wir wieder ein Unterprogramm:

Brech_Strahl(s,objnr,x,brech)

Der einzige Unterschied zu »Reflekt_Strahl« ist der Strahl »brech«. Der enthält den, nach dem optischen Brechungsgesetz an der Oberfläche von Objekt »objnr« gebrochenen Sehstrahl »s«.

Wieder »schaut« Raytrace, was in Richtung des Strahls »brech« liegt, indem es sich selber mit »brech« als Eingabe aufruft:

Statt einer »Aufrufkette« von »Raytrace_3« entsteht jetzt ein sogenannter »Aufrufbaum«. Bild 7.9 zeigt Spiegelung, Brechung und Aufrufbaum.

```
Raytrace_3(s,farb)
{
   Suche_Objekt(s,Objekt_nr,schnittpkt)
   Falls Objekt_nr>0
   dann  {
           schatt_str=Strahl von schnittpk zur Lichtquelle l
           Suche_Objekt(schatt_str,blockiert,schnittl)
           Falls blockiert=0
           dann Berechne_Farbe(s,Objekt_nr,schnittpkt,l,fl)
           sonst fl=schwarz
           Reflekt_Strahl(s,Objekt_nr,schnittpkt,r_strahl)
           Raytrace_3(r_strahl,f_spieg)
           Brech_Strahl(s,Objekt_nr,schnittpkt,b_strahl)
           Raytrace_3(b_strahl,f_brech)
           f=fl+f_spieg+f_brech
         }
   sonst f=hintergrund
}
```

Bild 7.8: Raytrace_3(s,farb)

Führen wir uns einmal vor Augen, wie »teuer« die einzelnen Effekte sind (angenommen, wir haben eine Szene, bei der der ganze Bildschirm mit Objekten gefüllt ist):

»Raytrace_0«, die einfachste Version, schickt pro Pixel einen Strahl los.

»Raytrace_1«, bei der Schatten berechnet wird, muß bereits pro Pixel zwei Strahlen untersuchen; einen Sehstrahl und einen Strahl zur Lichtquelle.

»Raytrace_2« muß dementsprechend drei Strahlen und »Raytrace_3« vier Strahlen pro Pixel untersuchen.

Die Zahlen für »Raytrace_2« und »_3« sind aber nur Minimalzahlen, denn wenn die Spiegel- bzw. Brechstrahlen wieder auf Objekte stoßen, werden von dort wieder jeweils zwei bzw. drei Strahlen ausgesandt.

Nehmen wir als Beispiel für »Raytrace_3« an, daß jeder Spiegel- und jeder Brechstrahl noch jeweils einmal auf ein Objekt stoßen, deren Nachfolger dann aber nicht mehr.

Wie man erkennen kann, müßten in diesem Fall pro Pixel 10 Strahlen verfolgt werden, das sind bei einer Bildschirmauflösung von 320*256 ca. 820000 Strahlen.

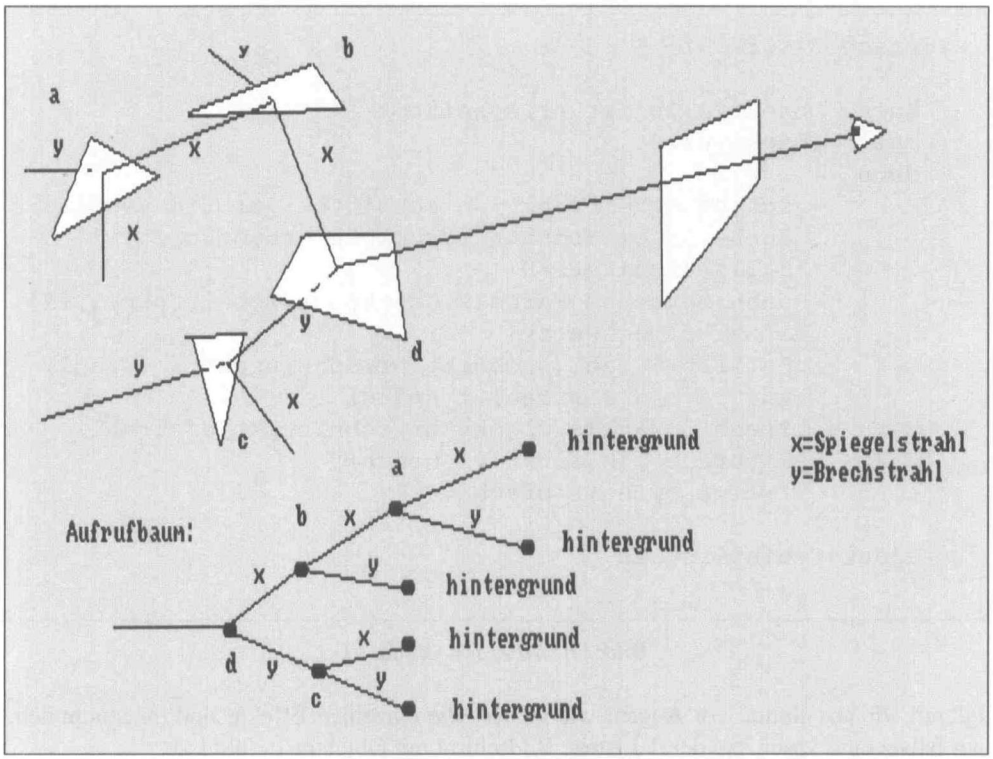


Bild 7.9: Spiegelung, Brechung und Aufrufbau

Das sind zwar hohe Zahlen, aber Sie lassen sich davon vielleicht nicht so schnell beeindrucken. Den Aufwand kann man auch erst übersehen, wenn man weiß, daß die meiste Rechenzeit beim Raytracing im Unterprogramm »Suche_Objekt« verbraucht wird, und das wird pro Strahl einmal aufgerufen. Wie die Prozedur nun im einzelnen aussieht, wird im nächsten Abschnitt behandelt. Vorher sei noch erwähnt, daß im allgemeinen der Aufwand nicht so hoch liegt wie im Beispiel beschrieben, weil der vorsichtige Designer nicht seine ganze Szene mit spiegelnden und durchsichtigen Objekten füllt, sondern die eher in der Minderzahl läßt. »Raytrace_3« überprüft selbstverständlich, – wieder ein Implementierungsdetail – ob ein Objekt, auf das es gestoßen ist, spiegelt und/oder durchsichtig ist. Es schickt dann einen Spiegel- bzw. Brechstrahl nur los, wenn es erforderlich ist.

Es mag aber wohl Zeitgenossen geben, die ihren Raytracer »in ein Spiegelkabinett stellen« und sich dann wundern, wenn die Rechenzeiten ins Astronomische wachsen. Das sind Extrembeispiele, aber man erkennt daraus wohl, daß Zeitangaben über die Geschwindig-

keit dieses oder jenes Raytracers fast nichts wert sind, wenn man das Bild dazu nicht gesehen hat und abschätzen kann, wie oft gespiegelt und gebrochen wurde.

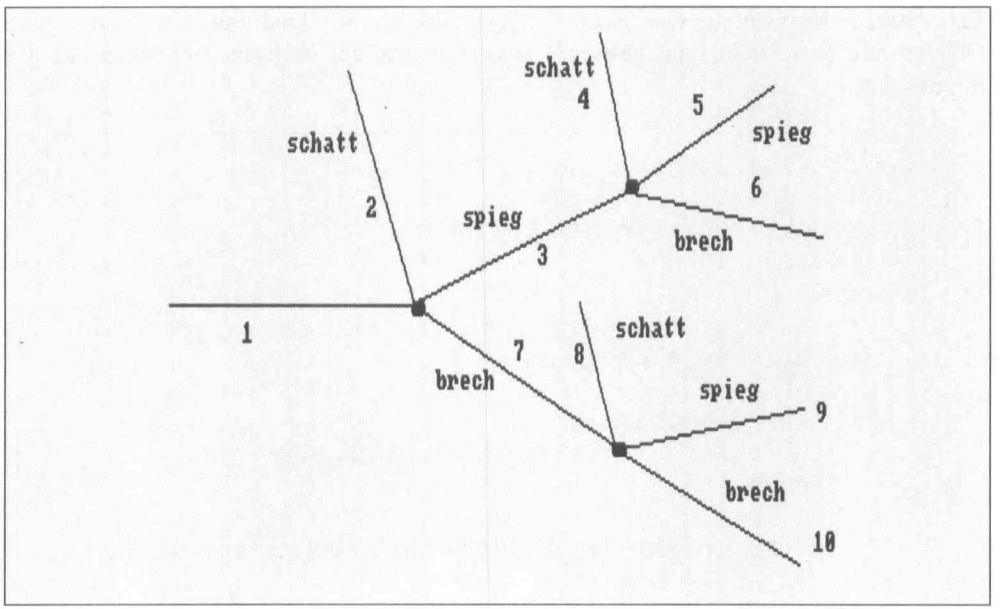


Bild 7.10: Die Anzahl von Strahlen bei Raytrace_tiefe 2

7.5 Suche_Objekt

Hier sind wir sozusagen im Herz des Raytracers. Die meiste Rechenzeit wird hier verbraucht, daher ist hier auch der große Ansatzpunkt für Verbesserungen. Wieso ist der Aufwand denn so groß?

Wie oben schon angeführt, haben wir für diese Prozedur als Eingabe einen Strahl und sollen für diesen Strahl herausfinden, ob er überhaupt ein Objekt trifft, und wenn ja, welches zuerst.

Wie man bei einem Strahl feststellt, ob er ein Objekt schneidet, ist wieder ein unwichtiges Detail, reine Geometrie bzw. Vektorrechnung. In Abschnitt 7.10.e sind einige Anmerkungen dazu festgehalten.

Stecken wir das Problem einfach wieder in eine handliche Funktion:

Strahl_trifft_Objekt(s,objnr,schnittpkt)

Sie bekommt als Eingabe den Strahl »s« und die Objektnummer »objnr«, und liefert 1, wenn »s« »objnr« schneidet. Zusätzlich wird dann in »schnittpkt« der Schnittpunkt von »s« mit dem Objekt geliefert. Wenn »s« nicht trifft, wird als Funktionswert 0 übergeben.

Eine einfache Realisierung von »Suche_Objekt« liegt auf der Hand. Man testet einfach alle Objekte mit dem Strahl, und bestimmt dasjenige, das am nächsten am Startpunkt des Strahls liegt.

```
Suche_Objekt(s,objnr,schnittpkt)
{
     schnitt=Punkt in der Verlängerung von s
     im»unendlichen«
     nr=0;

     für alle objekte
     {
        i=Nr des Objektes
        Falls Strahl_trifft_Objekt(s,i,s1)=1
         dann
         {
           Falls (s1 liegt näher am Startpunkt von s
              als schnitt)
           dann {
              nr=i
              schnitt=s1
                }
        }
     }
     schnittpkt=schnitt
     objnr=nr
}
```

Bild 7.11: Suche_Objekt(s,objnr,schnittpkt)

Anfangs wird die »nr« auf 0 gesetzt und »schnitt« auf einen Punkt gelegt, der in der Verlängerung von »s« im »Unendlichen« liegt. Das soll nur ausdrücken, daß »schnitt« jenseits aller Objekte ist, so daß der erste Schnittpunkt vom Strahl mit einem Objekt vor »schnitt« liegt. Danach wird eine Schleife über alle Objekte durchgeführt. Jedes Objekt wird mit dem Strahl auf Schnitt getestet. Wenn der Test erfolgreich ist und der Schnittpunkt näher am Startpunkt von »s« als der bisher nächste liegt, dann merkt man sich in »nr« die Objektnummer und in »schnitt« den Schnittpunkt.

So wird am Ende das Objekt geliefert, da zuerst von »s« geschnitten wird, oder 0, falls »s« gar kein Objekt trifft.

Man kann sich wohl davon überzeugen, daß »Suche_Objekt« genau das erfüllt, was »Raytrace_3« von ihr erwartet.

Aber man führe sich vor Augen, daß mit dieser Formulierung von »Suche_Objekt« jeder Strahl mit jedem Objekt auf Schnitt getestet werden muß. Die Strahlanzahl liegt, wie oben bereits erwähnt, bei einer Bildauflösung von 320*256 Pixeln irgendwo zwischen 80000 und 800000. Bei einer mittleren Szene aus 1000 Objekten müßten demnach zwischen 80 und 800 Millionen Schnittests durchgeführt werden, da ja jeder der Strahlen jeweils mit allen 1000 Objekten getestet werden muß.

Die Zeit für einen Schnittest zwischen einem Strahl und einem Objekt hängt stark vom Objekt und von der internen Speicherung des Objekts ab, aber mehr als einige 100 Schnittests pro Sekunde schafft der Amiga nicht. Wenn wir als Schätzwert 500 Tests pro Sekunde ansetzen, kommt man auf eine Rechenzeit von 40 Stunden, wenn man auf alle Effekte verzichtet (Raytrace_0) und bis zu 400 Stunden bei einer stark verspiegelten Szene.

So geht es also nicht, es sei denn, man beschränkt sich in seiner Szenengröße auf wenige Objekte.

Es gibt eine Reihe von Ansatzpunkten, um Rechenzeit einzusparen. Alle lassen sich hier gar nicht aufführen, aber der größte Brocken an Rechenzeit steckt, wie oben zu sehen war, in der »Suche_Objekt«-Prozedur. Und dort werden wir auch ansetzen und einiges herausholen.

Das Problem »finde in einer großen Objektmenge ein Objekt, das einen Strahl als erstes blockiert« kann man auch allgemein formulieren:

Finde in einer großen Datenmenge die Daten, die ein bestimmtes Kriterium erfüllen.

In der Informatik tauchen solche Probleme ständig auf, und es wurden viele Lösungen entwickelt, um dieses Suchproblem zu lösen. Den einfachsten Weg (die gesamte Datenmenge durchsuchen) haben wir ja schon kennengelernt.

Meistens werden die Daten nach irgendwelchen Kriterien sortiert, oder es werden zusätzliche Datenstrukturen geschaffen, die den Zugriff auf die eigentliche Datenmenge beschleunigen. Das klingt jetzt wenig anschaulich, aber an einem Beispiel wird es schnell deutlich:

Bei einem anderen Bilderzeugungsverfahren (siehe Kap. 8) werden die Objekte nach ihrer Entfernung zum Bildschirm sortiert. Danach werden die Objekte der Reihe nach auf den Bildschirm gezeichnet, die entfernteren zuerst. Da die näheren Objekte später gezeichnet werden, übermalen Sie die hinteren, wenn Sie sie (vom Betrachter aus gesehen) verdecken.

Ein Beispiel für eine zusätzliche Datenstruktur, die den Zugriff beschleunigen soll, bringt uns auch gleich zur ersten Verbesserung von »Suche_Objekt«:

Normalerweise sind ja die Objekte nicht gleichmäßig im Raum verteilt, sondern sie konzentrieren sich an bestimmten Stellen. Die Idee ist jetzt, diese Zusammenballungen von

Objekten mit unsichtbaren Kugeln zu umschließen, so daß die Objekte vollständig darin liegen.

Berücksichtigen wir das bei »Suche_Objekt«, so wird zuerst der Strahl mit einer Umgrenzungskugel auf Schnitt getestet. Schneidet er die nicht, so brauchen die Objekte, die in der Kugel sind, nicht mehr beachtet zu werden, da er Sie ja nicht schneiden kann. Nur wenn der Strahl die Kugel trifft, müssen die Objekte, die darin liegen, der Reihe nach auf Schnitt getestet werden.

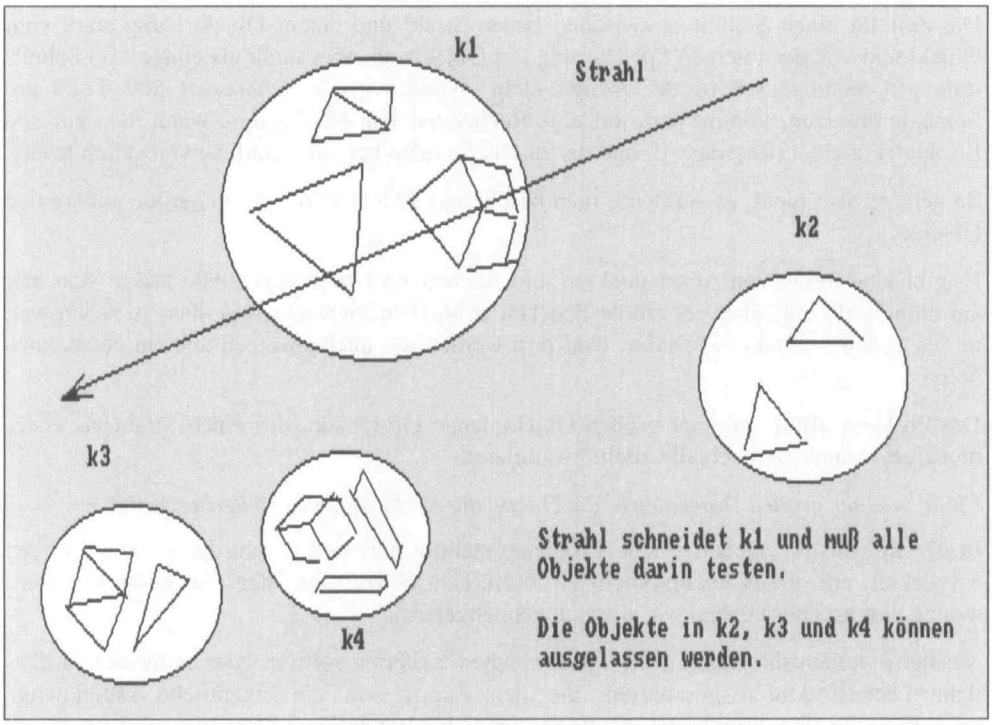


Bild 7.12: Objekte mit einem Kreis umschließen

Die Umgrenzungskugeln sind auf dem Bild nicht zu sehen. Sie werden von der Raytrace-Routine nicht beachtet. Die Kugeln sind die obenerwähnten zusätzlichen Daten, die den Zugriff auf die eigentlichen Daten (die Objekte) beschleunigen.

Damit können wir jetzt eine verbesserte Version von »Suche_Objekt« formulieren. Wir gehen jetzt davon aus, daß eine Reihe von Umgrenzungskugeln vorliegt, die jede eine Objektliste mit den Objekten enthält, die in ihr liegen. (Die folgenden Algorithmenansätze werde ich noch grober formulieren als bisher, da ich mich sonst in zu vielen Details verlieren würde.)

```
Suche_Objekt(strahl,nr)
{
 - Teste den Strahl mit allen Umgrenzungskugeln
 - falls der Strahl eine Um.Kugel schneidet
 dann
    {
     - Schneide den strahl mit allen Objekten in der
       Um.Kugel und merke dir das Objekt, das ihn zuerst
       schneidet
    }
 - nr = Nummer des Objekts, das den strahl zuerst
   schneidet oder 0 falls kein solches Objekt existiert
 }
```

Bild 7.13: Suche_Objekt(strahl,nr)

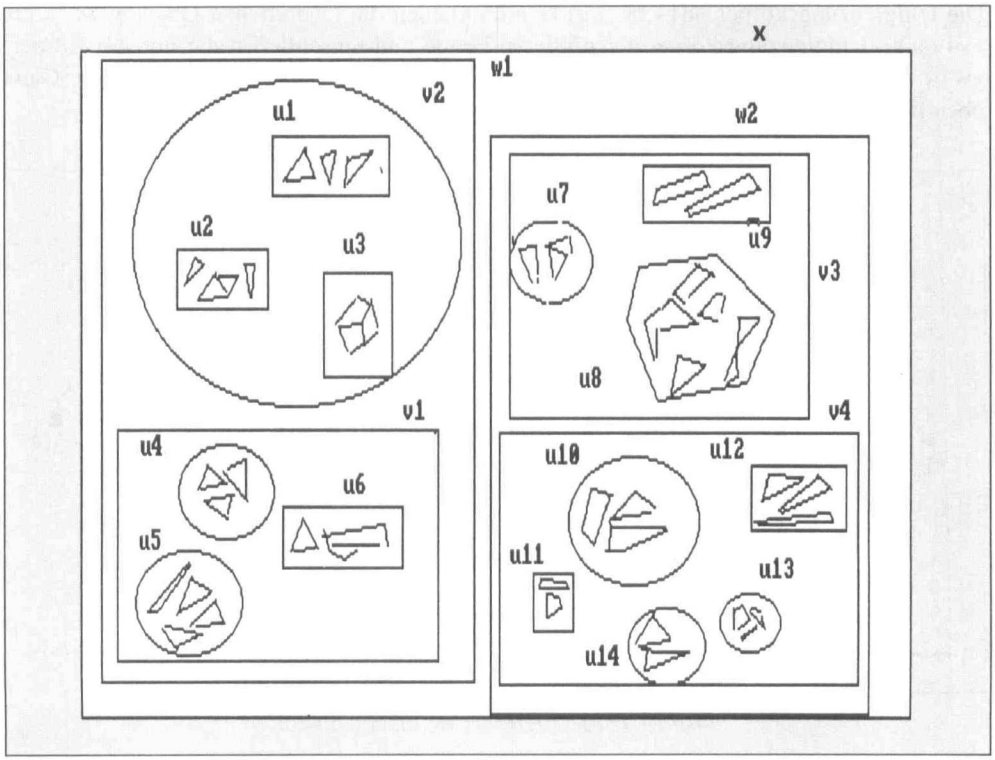


Bild 7.14: Neue Umgrenzungskörper

Zu dem Verfahren läßt sich folgendes sagen:

- Damit sich die Methode auch lohnt, darf es natürlich nicht zuviel Umgrenzungskugeln geben.
- Strahlen, die nur den Hintergrund treffen, müssen praktisch keine Objektschnittests mehr ausführen.
- Wenn sich die Objekte nicht geeignet zusammenballen oder ungünstig liegen (alle hintereinander), bringt diese Methode nicht viel Gewinn.
- Im allgemeinen versprechen Umgrenzungskugeln nur bei kleinen bis mittleren Szenen, die zudem klar gegliedert sind, Erfolg.

Allgemeine Umgrenzungskörper und Hierarchie

Man kann dieses Prinzip noch erweitern, indem man einmal noch andere Umgrenzungskörper zuläßt (Quader, Polyeder). Aus einer höheren Sicht können die Umgrenzungskörper aber auch wieder als Grundobjekte angesehen werden. Und die kann man dann erneut mit weiteren Umgrenzungskörpern umgeben.

Die Umgrenzungskörper »u1« bis »u14« umschließen die eigentlichen Objekte. »v1« bis »v4« sind Umgrenzungskörper der mittleren Ebene und umschließen die unteren Körper. »w1« und »w2« sind wieder eine Ebene höher und umschließen die »v«-Körper. Ganz oben liegt »x«, der alle Körper enthält.

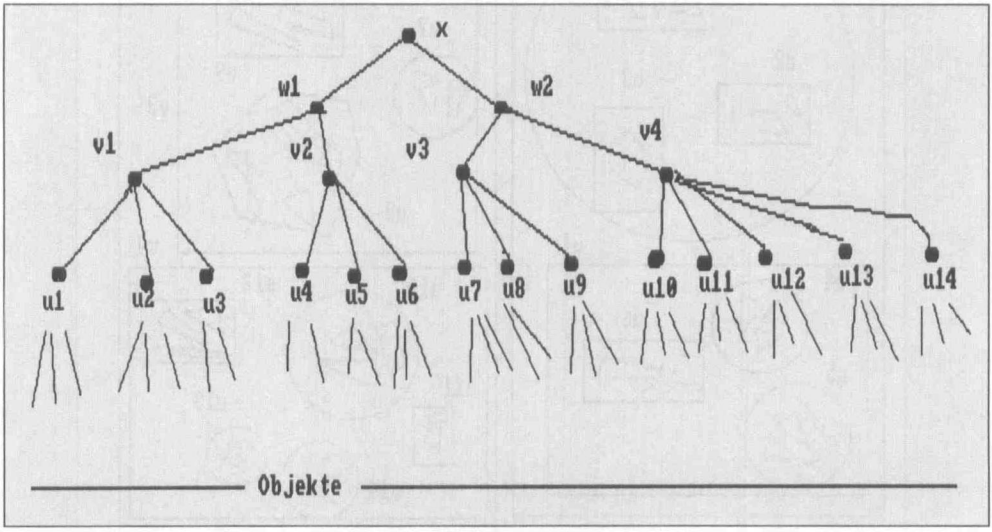


Bild 7.15: Die Hierarchie als Suchbaum

Dieses Prinzip vollzieht sich über mehrere Ebenen, bis schließlich am Ende ein Körper die ganze Szene umschließt. Damit ist ein »Suchbaum« entstanden, der eine Szene sehr fein einteilen kann. Bild 7.15 zeigt die Hierarchie als Suchbaum.

Bäume (7.10.f) sind in der Informatik ein häufig angewandtes Mittel, um Ordnung in große Datenmengen zu bringen. »Suche_Objekt« sähe bei Verwendung von Suchbäumen etwa folgendermaßen aus. »UG-Körper« sei ein irgendwie gearteter Umgrenzungskörper und enthält entweder eine Liste mit weiteren Körpern, die in ihm liegen, oder falls er auf der untersten Ebene liegt, eine Liste von Objekten, die er umschließt.

```
Suche_Objekt(strahl,nr,UG-Körper)

   {
      - falls UG-Körper weitere Körper enthält, dann
         {
            - rufe Suche_Objekt für alle in UG
              enthaltenen Unterkörper auf.
         }
              sonst
         {
            - Teste alle im UG-Körper enthaltenen
              Objekte auf Schnitt mit dem Strahl und
              merke dir in nr das Objekt, das zuerst
              getroffen wurde.
         }
    }
```

Bild 7.16: Suche_Objekt(strahl,nr,UG-Körper)

Dies mag jetzt umständlich aussehen, aber es spart tatsächlich in den meisten Fällen sehr viele Objektschnittests ein. Gegeben sei eine Szene mit ca. 1000 Objekten. Man hat nun 50 Umgrenzungskörper so angeordnet, daß sie jeweils 20 Objekte enthalten. Dann sind noch mal in einer höheren Ebene weitere 10 Umgrenzungskörper so angeordnet, daß sie je 5 der unteren Umgrenzungskörper enthalten.

Gehen wir jetzt mit einem Strahl in die Szene hinein (mittels »Suche_Objekt«), so wird der Strahl zuerst mit den 10 Umgrenzungskörpern der oberen Ebene getestet. Von dem getroffenen Körper werden nun alle darin enthaltenen Körper der unteren Ebene getestet, also nochmal 5 Tests. Schließlich muß der Strahl mit den Objekten, die in dem getroffenen unteren Umgrenzungskörper liegen, auf Schnitt getestet werden, um endlich das Objekt zu finden, das der Strahl zuerst trifft. Anstatt 1000 Objektschnittests auszuführen, wie bei der einfachsten Version von »Suche_Objekt«, müssen jetzt nur noch 20 Objekte überprüft werden und zusätzlich 15 Schnittests mit Umgrenzungskörpern. Die sind zwar normaler-

weise etwas aufwendiger als einfache Objektschnittests, aber liegen doch ungefähr in der gleichen Größenordnung. Dieses Verfahren führt also, so umständlich es auch sein mag, doch zu einer gewaltigen Einsparung.

Diese Methode hat jedoch auch Nachteile, die im folgenden Abschnitt behandelt werden.

7.6 Die Gittermethode

Bei der Umgrenzungskörpermethode wurden bereits ein Menge Schnittests mit Objekten eingespart. Das wurde allerdings mit einem zusätzlichen Aufwand erkauft, mußte der Strahl doch mit vielen Umgrenzungskörpern geschnitten werden. Und diese Tests sind leider auch nicht gerade billig. Im Beispiel hatten wir am Ende immer noch mindestens 20 Objekt- und 15 Körperschnittests pro Strahl. Wir können diese Zahl natürlich noch senken, indem wir die Körper der unteren Ebene, die die Objekte enthalten, so anordnen, daß jeder nur noch zum Beispiel vier Objekte enthält. Nur erhöht sich die Körperanzahl dadurch gewaltig. Wir erkaufen uns die Einsparungen bei den Objektschnittests durch einen immer höheren Aufwand, um mit dem Strahl die Umgrenzungskörper zu testen.

Ein weiterer Nachteil der letzten Methode ist die Definition der Umgrenzungskörper. Wie ordne ich sie so an, daß mein Aufwand insgesamt minimal wird? Der Aufwand, um diese Einteilung der Objekte zu den Körpern vorzunehmen, wächst mit zunehmender Objektanzahl stärker an als der Raytrace-Aufwand, so daß irgendwann kein Gewinn mehr zu erzielen ist.

Was wir bräuchten, wäre ein Verfahren, um die Objekte sehr fein in verschiedene Bereiche einzuteilen, das aber schnell feststellen kann, welcher Bereich vom Strahl getroffen wird. Die Gittermethode erfüllt genau diese Ansprüche.

1) Teile den Raum, den die Objekte füllen, mit einem dreidimensionalen Gitter in lauter gleichmäßige Zellen ein.

2) Stelle für jede der Zellen fest, welche Objekte in ihr liegen oder sie zumindest schneiden. Danach existiert zu jeder Zelle eine Liste mit den Objekten, die in ihr sind (oder eine leere Liste, wenn auch die Zelle leer ist).

Die beiden ersten Schritte sind in Bild 7.17 verdeutlicht.

Beim Raytracing müssen wir nur noch ermitteln, durch welche Zellen unser Strahl läuft und ihn dann mit den Objekten auf Schnitt testen, die in den Listen der jeweiligen Zellen enthalten sind.

Jetzt fragen Sie sich vermutlich, wo denn hier der Unterschied zur Körpermethode ist; die Zellen bei der Gittermethode entsprechen doch den Umgrenzungskörpern. Im Prinzip ist das richtig, nur haben die Zellen einen kleinen, aber wichtigen Vorteil:

Die Zellen sind Teil eines Gitters, und daher regelmäßig und alle gleich groß. Bei der Körpermethode mußte ein Strahl immer mit allen Körpern getestet werden, hier braucht man das mit den Zellen nicht zu machen. Man ermittelt lediglich die Zelle, in der der Startpunkt

des Strahls liegt. Danach kann man sich Zelle für Zelle durch das Gitter hangeln, genau in der Richtung, die der Strahl hat. Dies geht zudem sehr »billig«. Denn zum Durchqueren des Gitters wird ein sogenannter »Vektorgenerator« verwendet. Er arbeitet genauso wie ein Vektorgenerator zum Zeichnen von Linien auf einem Pixelbildschirm. Der muß ja auch die Pixel ermitteln, die eine imaginärer Strich berühren würde, und diese Pixel werden dann mit einer Farbe gefüllt. Ein solcher Vektorgenerator ermittelt das jeweils nächste Pixel lediglich durch eine Subtraktion und einen Vergleich, durch den er feststellt, in welcher Richtung das nächste Pixel liegt.

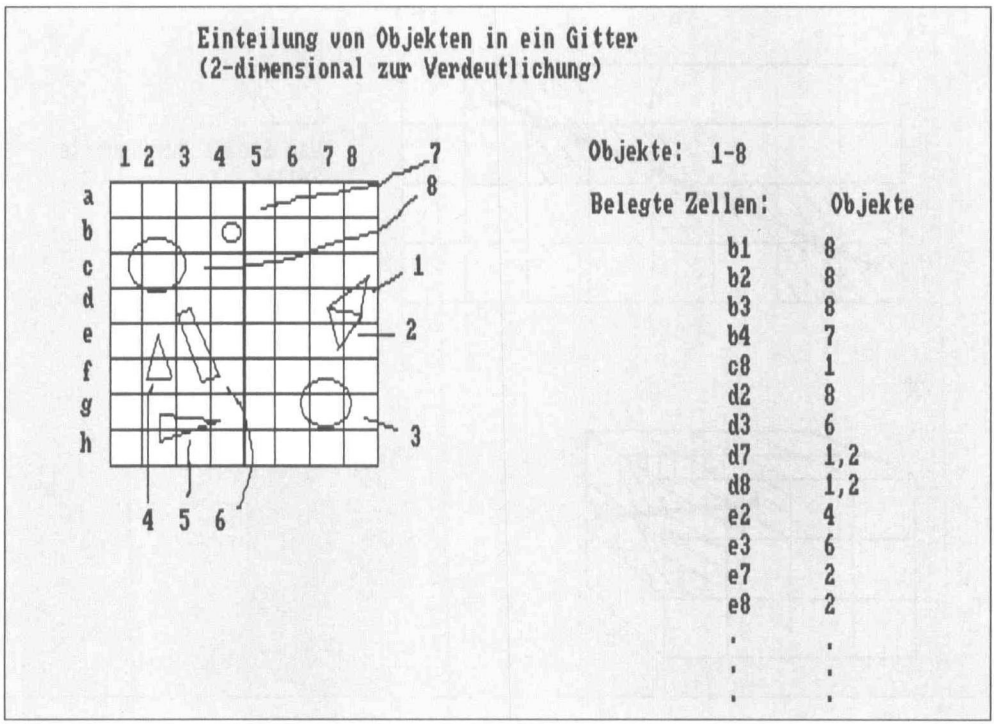


Bild 7.17: Einteilung von Objekten in ein Gitter

Bei unserem Gitter entsprechen die Zellen den Pixeln beim Bildschirmvektorgenerator. Allerdings ist das Gitter dreidimensional, deswegen muß ein dreidimensionaler Generator verwendet werden. Das ist auch kaum ein größerer Aufwand, es läuft im Prinzip auf zwei parallel laufende 2-D-Generatoren hinaus. Es müssen dabei pro Zelle je zwei Subtraktionen und Vergleiche angestellt werden, um die nächste Zelle zu ermitteln, die der Strahl durchquert.

Wo also bei der Körpermethode pro Körper ein aufwendiger Test mit dem Strahl stattfindet, sind hier pro Zelle, die der Strahl durchquert, nur zwei Subtraktionen und Vergleiche (zudem mit Integer-Arithmetik) nötig. Außerdem sorgt der Vektorgenerator dafür, daß immer nur die Zellen zu testen sind, die der Strahl auch durchquert, und keine einzige mehr.

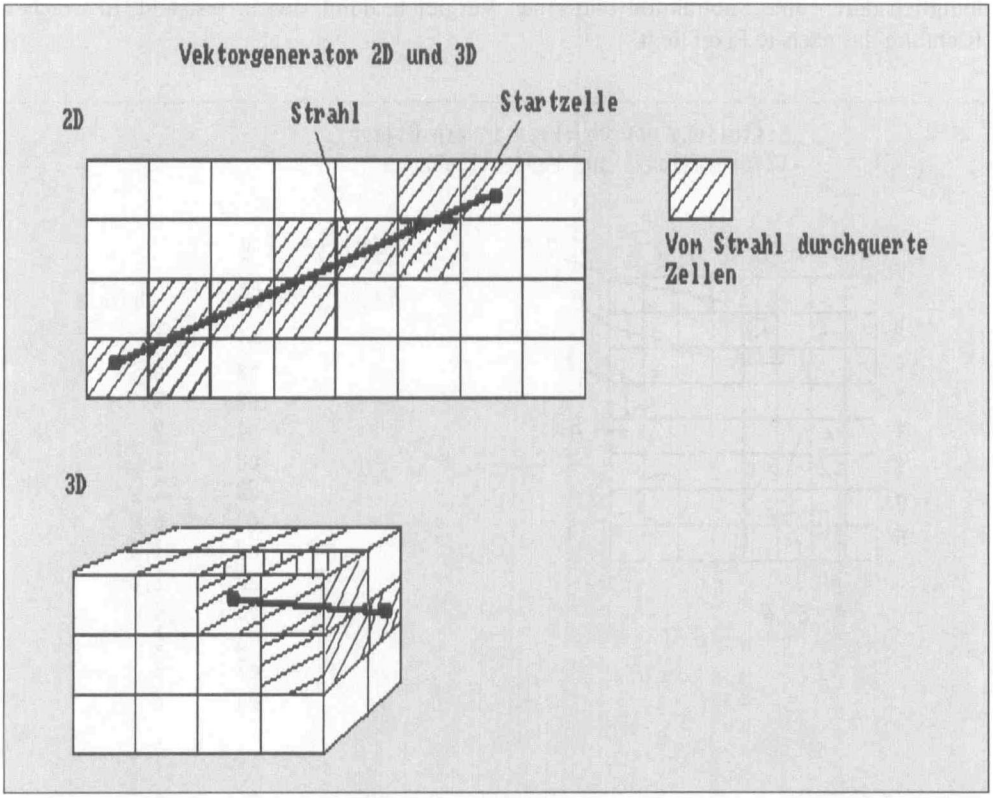


Bild 7.18: Vektorgenerator 2D und 3D

Da man so schnell die Zellen durchqueren kann, macht man natürlich das Gitter so fein wie möglich. Dadurch enthalten die einzelnen Zellen weniger Objekte, was wiederum zur Einsparung von Objektschnittests führt.

Die »Suche_Objekt«-Prozedur sähe mit der Gittermethode ungefähr so aus:

```
Suche_Objekt(Strahl,objekt)

{
   - Berechne die Startzelle des Strahls
   - Solange, bis ein Objekt getroffen ist, oder das
     Gitter vollständig durchquert wurde
   {
      - Falls die Objektliste der aktuellen Zelle nicht
        leer ist, dann
          - Schneide den Strahl mit allen Objekten
            der Zelle und merke dir das Objekt, das
            der Strahl als erstes trifft.
        Sonst
          - Springe zur nächsten Zelle, die der
            Strahl durchquert
   }
}
```

Bild 7.19: Suche_Objekt(Strahl,objekt)

Die Gittermethode ist aber in dieser Version auch noch nicht für alle Szenen geeignet.

7.7 Der Gitterbaum

Einfache Gitter sind nicht für alle Szenen geeignet. Sie würden sich am besten bei Szenen auswirken, bei denen die Objekte gleichmäßig über einen bestimmten Raumbereich verteilt sind. Leider ist das meistens nicht der Fall. Vielmehr ballen sie sich an bestimmten Punkten der Szene und erstrecken sich entlang von Oberflächen. Das führt aber dazu, daß die meisten Zellen leer sind, während im schlimmsten Fall sich in einigen Zellen sehr viele Objekte tummeln. Falls der Betrachter nun seinen Blick ausgerechnet auf solch eine Objektzusammenballung richtet, durchqueren die Strahlen die Zellen, die so dicht mit Objekten belegt sind, und damit haben wir wieder die gleiche Situation wie am Anfang, nämlich zuviele Objektschnittests pro Strahl. Oder die Objekte liegen sehr weit auseinander, der Zwischenraum muß aber mit Zellen gefüllt werden, was zu einer sehr großen Zellanzahl führt, und das gibt Speicherprobleme.

Dummerweise kann man das Gitter nicht beliebig fein machen. Da es ein dreidimensionales Gitter ist, steigt die Zellanzahl mit der dritten Potenz zur Zelldichte an. (Ein 8*8*8-Gitter hat achtmal so viele Zellen wie ein 4*4*4-Gitter.)

Die Lösung für dieses Problem ist der Gitterbaum. Es ist eine Verbindung der hierarchischen Körpermethode mit dem Gitterverfahren. Das bekannteste und am häufigsten verwendetste Beispiel ist der »Octtree«. Er beginnt in der obersten Stufe mit acht Zellen (2*2*2). Jede dieser Zellen kann wiederum in weitere acht Zellen unterteilt werden, was sich dann über viele Ebenen vollziehen kann.

Für das Raytracing teilt man die Objekte zuerst den acht Zellen der obersten Stufe zu. Sobald in einer Zelle zuviele Objekte sind, wird diese wieder in acht Zellen aufgeteilt, für die jeweils genauso verfahren wird. Die »Octtree«-Struktur paßt sich so der jeweiligen Szene dahingehend an, daß sie dort sehr feine Zellen erzeugt, wo sich auch viele Objekte befinden, während in Regionen, wo nur wenige oder gar keine Objekte sind, die Zellen sehr groß sind. An der zweidimensionalen Variante des Octtrees, dem »Quadtree«, wo es jeweils nur vier Zellen gibt, kann man gut demonstrieren, wie sich das Prinzip auswirkt.

Die Bereiche, in denen in der Szene wenig los ist, das heißt, sich wenige Objekte befinden, werden beim Raytracing mit dem Vektorgenerator schnell durchquert. Dort, wo sich viele Objekte befinden, sind die Zellen hingegen sehr klein und enthalten demnach nur kurze Objektlisten. Insgesamt muß der Strahl nur mit sehr wenigen Strahlen getestet werden, praktisch auch nur mit solchen, an die der Strahl sehr nahe herankommt.

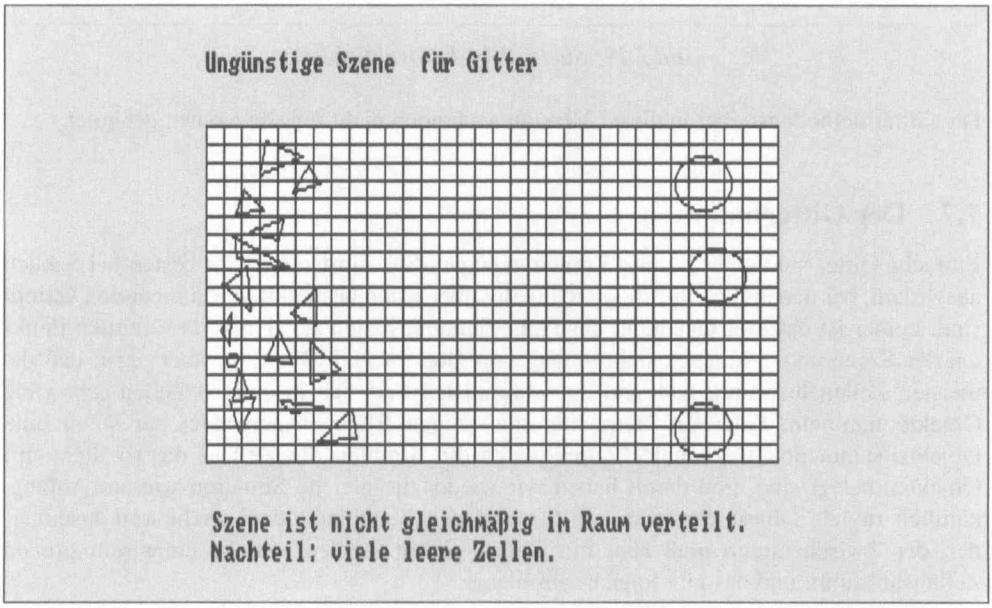


Bild 7.20: Ungünstige Szene für Gitter

Ein weiterer Vorteil liegt in der Erzeugung des »Octtrees«. Das Verfahren zur Erzeugung desselben aus einer Szene ist relativ einfach und ist nicht an bestimmte Szenentypen gebunden. Das Verfahren verarbeitet prinzipiell alle Typen von Szenen.

Der Octtree ist im Prinzip schon das Verfahren, das auch von Reflections benutzt wird. Er hat pro Gitter acht Zellen (2*2*2); dies lohnt allerdings kaum den Einsatz des Vektorgenerators. Der ist ja aus einem Gitter (2*2*2) Zellen schon wieder heraus, ehe er richtig angefangen hat. Da der Vektorgenerator ein sehr billiges Mittel zur Wahl der nächsten Zelle ist, kann die Zellanzahl pro Gitter schon etwas höher gewählt werden. Reflections verwendet daher Gitter mit je 512 Zellen (8*8*8). Jede der Zellen, die zuviele Objekte enthält, wird erneut in 512 weitere Zellen aufgeteilt, was rasch zu einer sehr feinen Aufteilung der Szene führt.

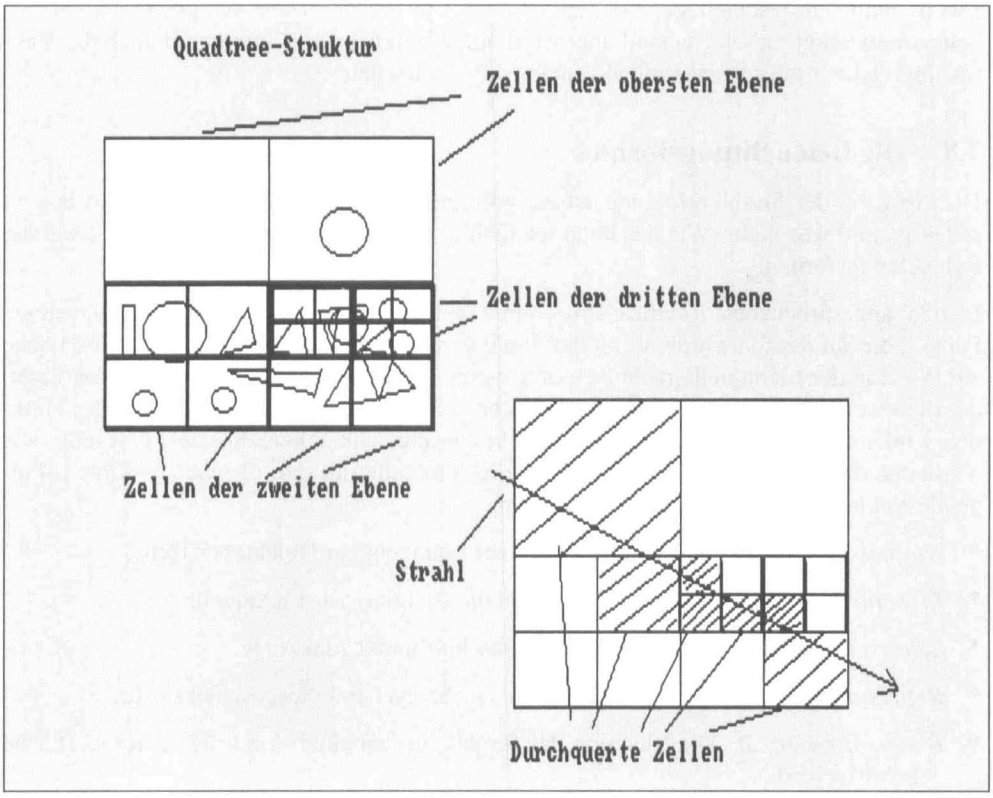


Bild 7.21: Quadtree-Struktur

Die Einsparungen durch das Verfahren sind in der Tat beträchtlich. In »Beams« muß ein Strahl im Durchschnitt nur noch zwischen zwei und acht Objekte auf Schnitt testen, unab-

hängig von der Objektanzahl in der Szene. Dafür entsteht natürlich ein gewisser Aufwand zur Erzeugung des Gitterbaums. Dies wird bei Reflections durch das Programm »Grid« automatisch vor dem Start von »Beams« vorgenommen. Die Rechenzeit für »Grid« ist allerdings verglichen mit »Beams« immer noch zu vernachlässigen, denn »Grid« rechnet im Bereich von Minuten, während sich »Beams« schon eher Stunden beschäftigt (für volle Bilder). Schwerer wiegt da schon der Speicherverbrauch. Bei großen Szenen erzeugt »Grid« eine Unmenge von Zellen. Davon sind die meisten zwar leer, aber die belegten sind immer noch recht viele. Und für jede belegte Zelle muß eine Liste mit Objektnummern angelegt werden. Trotz einer möglichst platzsparenden Speicherung (leere Zellen belegen nur ein Bit, doppelte Listen werden nur einmal abgespeichert etc.) ist der Umfang der Gitterbaum-Datenstruktur zwischen zwei- und fünfmal so groß wie die eigentlichen Objektdaten.

Das ist schon ein Nachteil, denn es schränkt den Umfang der Szene ein. Der Geschwindigkeitsvorteil wiegt diesen Nachteil aber wohl auf, ermöglicht der Gitterbaum doch das Raytracing relativ großer Szenen mit akzeptablen Rechenzeiten.

7.8 Die Beleuchtungsformel

Die Aufgabe der Strahlverfolgung ist es, grob gesagt, festzustellen, was von wo aus zu sehen ist und was nicht. Wie das dann letztlich in Farben umgesetzt wird, das erledigt die Beleuchtungsformel.

In den angesprochenen Raytrace-Prozeduren gab es immer eine Funktion »Berechne_ Farbe«, die aufgerufen wurde, wenn der Strahl von einem Punkt auf einer Objektoberfläche zur Position der Lichtquelle nicht blockiert wurde. Sie berechnet die Farbe bzw. das Licht, das dieser eine Punkt auf der Objektoberfläche von der Lichtquelle bekommt. In der Funktion wird sehr viel mit Vektoren und Winkeln zwischen ihnen gerechnet. Es gibt sehr viele Varianten der Beleuchtungsformel, aber sie rechnen alle mit den gleichen Vektoren. Folgende Vektoren gehen in die Berechnung ein:

- *Normalenvektor N:* Der Vektor steht immer senkrecht zur Objektoberfläche.
- *Lichtquellenvektor Lj:* Der Vektor enthält die Richtung zur Lichtquelle Nr.j.
- *Sehstrahl V:* Der Vektor zeigt vom Objektschnittpunkt zum Auge.
- *Reflektionsvektor R:* Die Richtung, in die der Spiegelstrahl losgeschickt wird.
- *Brechungsvektor B:* Die Richtung des Strahls, der an einer durchsichtigen Oberfläche gebrochen wird.

Wie Licht bzw. Farbe intern im Rechner gehandhabt wird, ist recht einfach. Eine Farbe besteht aus Rot-, Grün- und Blauanteil. Beim Amiga kann man beispielweise jedem Farbanteil Werte zwischen 0 und 15 zuteilen. Aus der Kombination der Rot-, Grün- und Blauanteile entsteht dann der Farbeindruck. Um einen photorealistischen Bildeindruck zu gewinnen, sind aber je 256 Abstufungen für Rot, Grün und Blau nötig.

Um es vorwegzunehmen: Eine vollkommen naturgetreue Simulation des Lichtverhaltens ist (noch) so gut wie unmöglich, da spielen noch zu viele »Unendlichkeiten« mit, die man nur grob vereinfachend im Rechner behandeln kann. Beim Raytracing war das schon nötig, als aus der unendlichen Menge der Lichtstrahlen, die eigentlich vom Auge ausgehen, nur eine relativ kleine Anzahl herausgepickt wurde. Auch bei der Berechnung der Farbe müßte man eigentlich davon ausgehen, daß jeder Punkt auf einem Objekt von überall im Raum Licht abbekommt, also nicht nur von der Lichtquelle, sondern auch viel Streulicht, das von anderen Objekten ausgeht.

Das realistische Lichtverhalten versucht man modellhaft nachzubilden. Es gibt da viele Varianten, aber meist geht man von einem Grundmodell aus:

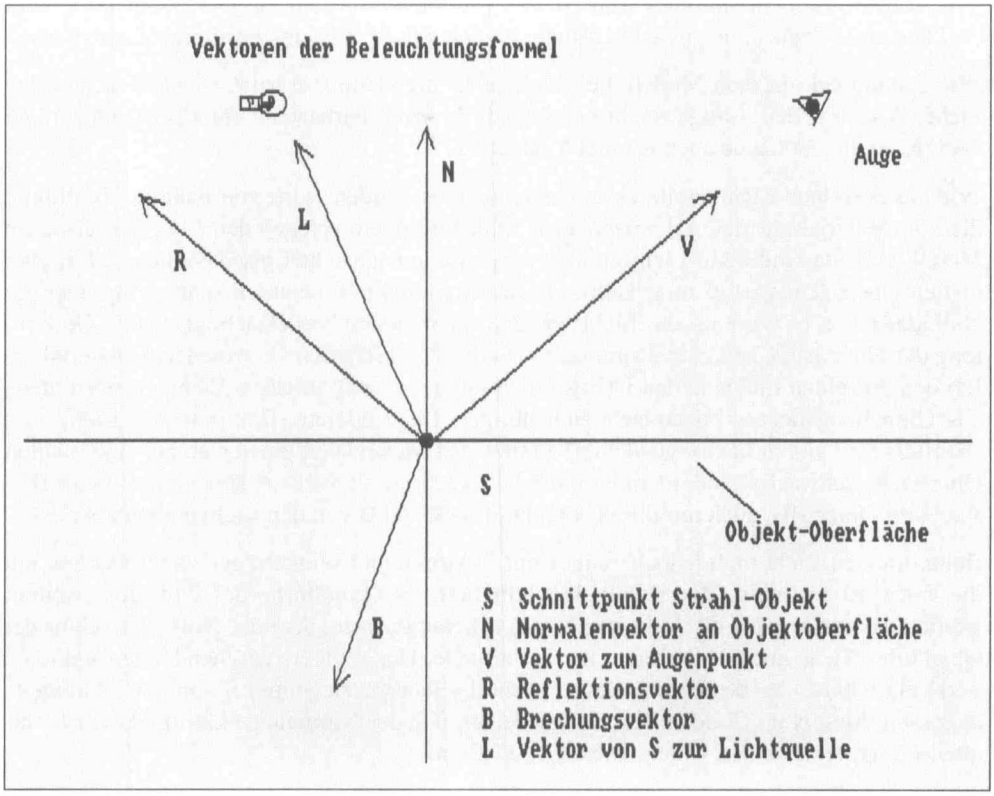


Bild 7.22 zeigt die Vektoren der Beleuchtungsformel

Das Licht, das an einem Oberflächenpunkt eines Objektes erscheint, setzt sich aus mehreren Anteilen zusammen.

- *Direktes Licht:* Das ist das Licht, das direkt von den Lichtquellen kommt. Hier muß auch zuvor ein Strahl zur Lichtquelle geschickt werden, um festzustellen, ob die Lichtquelle nicht verdeckt ist.
- *Indirektes Licht:* Indirektes Licht kommt von anderen Objekten. Um das zu ermitteln, wird beim Raytracing ein Strahl in die Reflektionsrichtung geschickt bzw. in die Brechungsrichtung, falls das Objekt durchsichtig ist. Die Farbe dieses Strahls wird dann mitverrechnet.
- *Umgebungslicht:* Umgebungslicht ist überall in der Szene gleichmäßig verteilt. Damit soll das Streulicht simuliert werden, das von allen Objekten ausgeht. Umgebungslicht kann einer Szene eine gewisse Grundhelligkeit geben oder eine bestimmte Grundfarbe.

Wie genau man mit dem Modell die richtigen Lichtverhältnisse trifft, ist oft Geschmackssache. Aber für den Amiga reicht es allemal, da seine Farbanzahl für realistische Bilder noch zu gering ist (siehe auch Kapitel 5, SHOW).

Wie die einzelnen Lichtanteile zusammengerechnet werden, wird von Faktoren bestimmt, die von Material zu Material verschieden sind. Bei einem spiegelnden Material, etwa ein Metall, wird das indirekte Licht zum Beispiel stärker gewichtet als das direkte. Bei eher matten Oberflächen ist es umgekehrt. Da ist der indirekte Lichtanteil sehr gering oder gar Null, das heißt, es wird nur das Licht von den Lichtquellen berücksichtigt. Durch Gewichtung der einzelnen Lichtanteile simuliert man die Eigenschaften verschiedener Materialien. Bei den einzelnen Lichtanteilen ist eigentlich nur etwas zum direkten Licht zu sagen, denn das Umgebungslicht wird einfach zum übrigen Licht addiert. Das indirekte Licht wird ebenfalls (mit einem Faktor gewichtet) addiert (bei Metall stark gewichtet, bei einer matten Oberfläche schwach). Außerdem ist indirektes Licht, das Objekt A abbekommt (zum Beispiel von Objekt B), auch nur direktes Licht, das Objekt B von den Lichtquellen abkriegt.

Beim direkten Licht muß jedoch einiges mit Vektoren und Winkeln gerechnet werden, um die Farbe zu ermitteln. Zuerst soll die Helligkeit der Grundfarbe des Objektes ermittelt werden. Hierzu werden die Vektoren N und L herangezogen, also der Normalenvektor der Objektoberfläche und die Richtung zur Lichtquelle. Der Winkel zwischen beiden Vektoren wirkt als Faktor, der bestimmt, wieviel Licht der Oberflächenpunkt S von der Lichtquelle bekommt. Sind N und L identisch, was bedeutet, daß der Normalenvektor direkt zur Lichtquelle zeigt, erhält S den vollen Anteil (Bild 7.23 a).

Sind N und L dagegen senkrecht zueinander, so bekommt S kein Licht von der Quelle, da deren Lichtstrahlen parallel an S vorbeiziehen (Bild 7.23 b). Alle Winkel von N und L, die zwischen 0 und 90 Grad liegen, werden entsprechend verrechnet. Man nimmt für die Berechnung des Faktors die Cosinus-Funktion, da die bei 0 Grad als Faktor 1 liefert und bei 90 Grad 0.

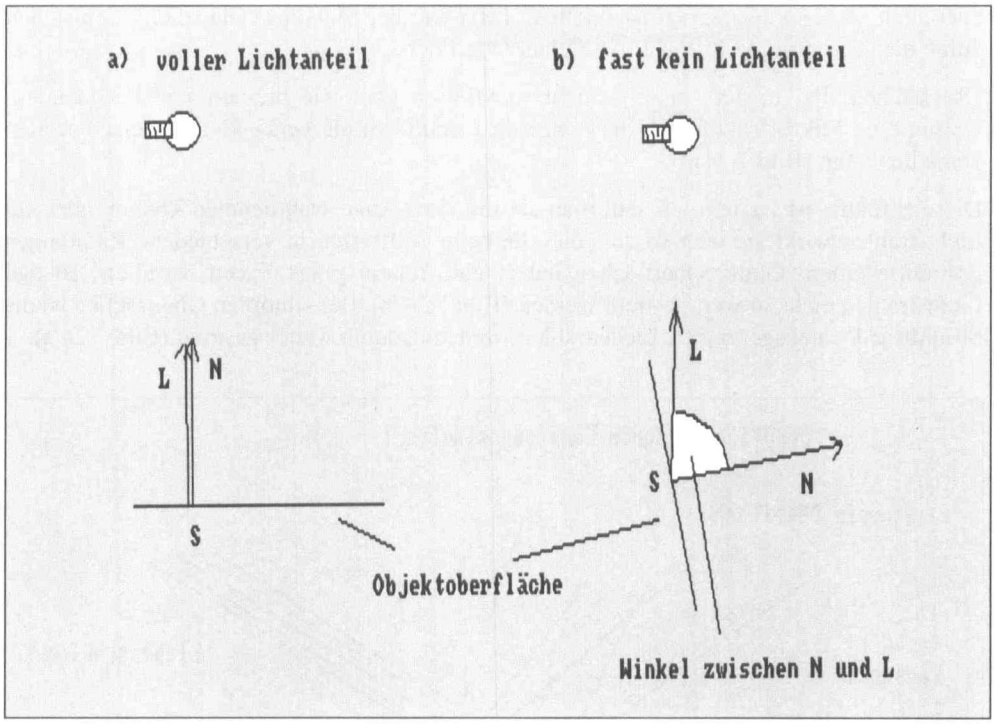


Bild 7.23a/b: Vektoren und Lichtanteil

Die Farbe errechnet sich demnach folgendermaßen aus Grundfarbe des Objektes, der Lichtfarbe und den Vektoren N und L:

Faktor = Cosinus des Winkels zwischen N und L
*Farbe.rot = Grundfarbe.rot * Lichtfarbe.rot * Faktor*
*Farbe.grün = Grundfarbe.grün * Lichtfarbe.grün * Faktor*
*Farbe.blau = Grundfarbe.blau * Lichtfarbe.blau * Faktor*

Die jeweiligen Farbanteile haben hier immer Werte zwischen 0.0 und 1.0.

Zu der Farbe kommt aber jetzt noch das Glanzlicht hinzu. So gut wie jede Oberfläche bekommt ein Glanzlicht, wenn sie beleuchtet wird. Beobachten Sie einmal beliebige Gegenstände, drehen Sie sie im Licht hin und her oder verändern Sie Ihren Blickpunkt. Das Glanzlicht wandert mit, woraus zu schließen ist, daß es von dem Beobachterstandpunkt abhängig ist. Glanzlichter unterscheiden sich aber auch stark – je nach Oberflächeneigenschaft. Glatte Oberflächen, wie Metall oder Glas oder zum Beispiel lackiertes Holz, haben ein ganz scharf ausgeprägtes Glanzlicht. Matte oder stumpfe Materialien hingegen erzeugen ein relativ weiches Glanzlicht, das sich über einen größeren Bereich erstreckt,

aber auch nicht so ausgeprägt ist (nicht so hell) wie bei Metallen. Glanzlichter entstehen durch die geometrische Mikrostruktur einer Oberfläche.

Oberflächen sind in der Natur ja nicht so klinisch glatt wie hier im Rechner, sondern weisen eine Mikrostruktur auf, die man sich vereinfacht als lauter kleine ebene Facetten vorstellen kann (Bild 7.24).

Diese Struktur ist zu fein, als daß man sie mit dem Auge wahrnehmen könnte, aber auf Lichtstrahlen wirkt sie sich so aus, daß die beim Auftreffen in verschiedene Richtungen gestreut werden. Glatte Oberflächen haben eine relativ glatte Facettenstruktur, so daß Lichtstrahlen nicht so weit gestreut werden (Bild 7.24 b). Bei stumpfen Oberflächen ist die Struktur jedoch ausgeprägter; Lichtstrahlen werden dadurch weiter gestreut (Bild 7.24 a).

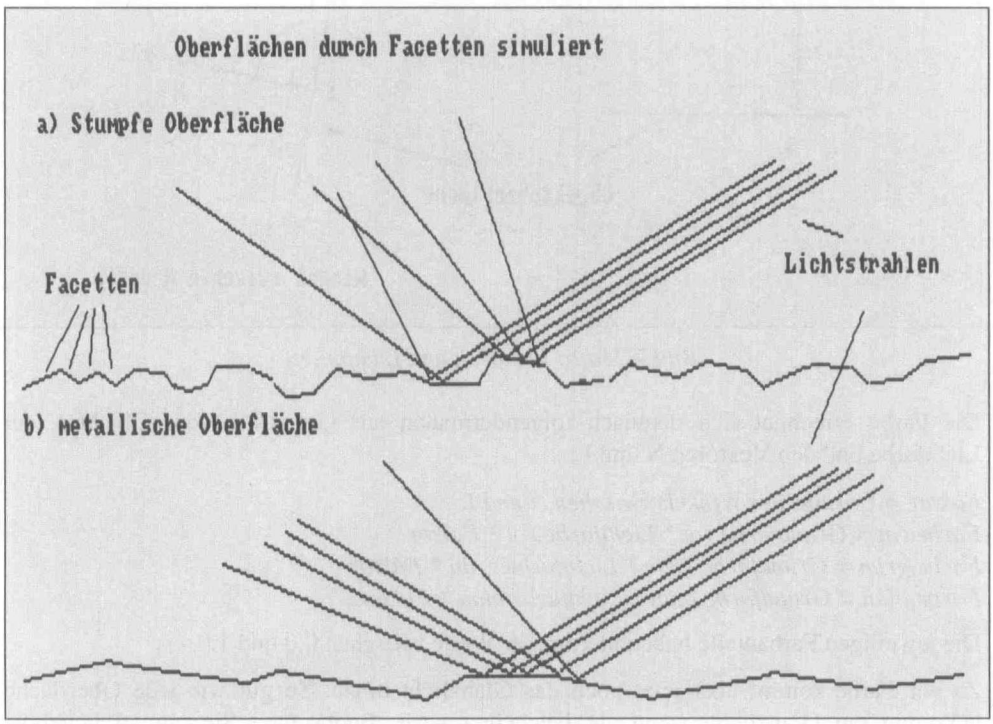


Bild 7.24a/b: Oberflächen durch Facetten simuliert

Das Glanzlicht, das wir sehen, ist der Anteil an Lichtstrahlen, die direkt in unser Auge reflektiert werden. Dieser Anteil ist bei glatten Oberflächen größer als bei stumpfen, was zu einem ausgeprägteren Glanzlicht führt.

Im Rechner wäre es selbstverständlich zu aufwendig, sich eine Oberfläche aus den Facetten zusammenzusetzen. Man simuliert sie nur durch die Glanzlichtfunktion.

Hierfür brauchen wir noch den Vektor R, den Reflektionsvektor. Das Glanzlicht ist dann am ausgeprägtesten, wenn der Strahl von der Lichtquelle genau ins Auge reflektiert wird. Das ist dann der Fall, wenn der Vektor R auf die Lichtquelle zeigt. Man rechnet hier also mit dem Winkel zwischen den Vektoren L und R.

Das Glanzlicht ist dann am hellsten, wenn R und L identisch sind (Bild 7.25 a) bzw. ist am schwächsten, wenn R und L fast entgegengesetzt zeigen (Bild 7.25b). Man könnte hier wieder mit dem Cosinus rechnen, aber das entspräche nur einem ganz weichen Glanzlicht. Man braucht hierfür eine Funktion, die bei 0 Grad ihr Maximum erreicht, sehr schnell abfällt und spätestens bei 180 Grad den Wert 0 hat. In Reflections hat beispielsweise jede Oberfläche ihre eigene Glanzlichtfunktion als Tabelle mit gespeichert. In Bild 7.26 sind einmal zwei Glanzlichtfunktionen für eine glatte und eine stumpfe Oberfläche gezeigt.

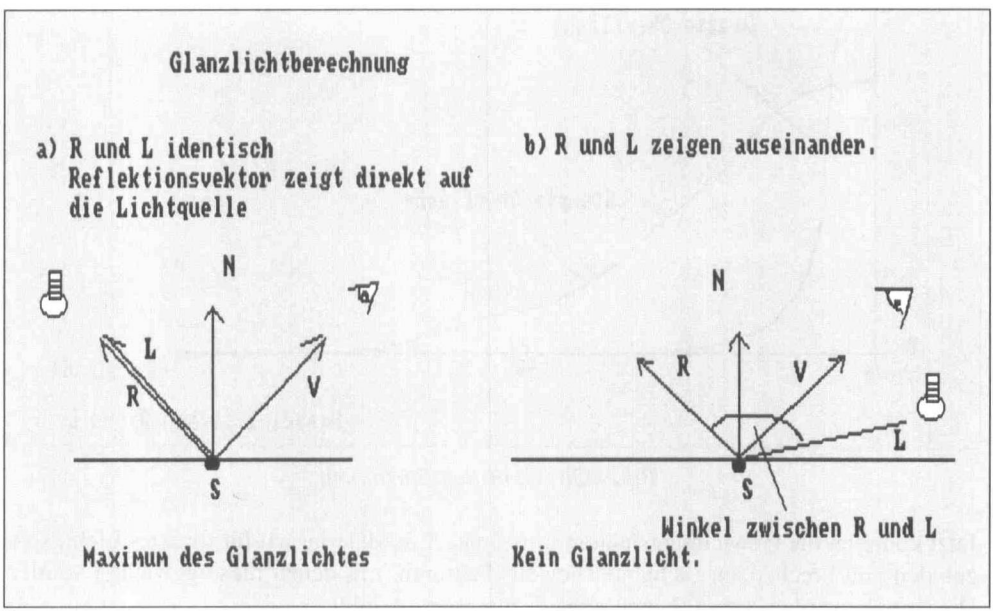


Bild 7.25a/b: Glanzlichtberechnung

Das Glanzlicht errechnet sich damit folgendermaßen:

winkel	= Winkel zwischen R und L
glanzlicht.rot	= Glanzfunktion(winkel) * Lichtfarbe.rot
glanzlicht.grün	= Glanzfunktion(winkel) * Lichtfarbe.grün
glanzlicht.blau	= Glanzfunktion(winkel) * Lichtfarbe.blau

Das Glanzlicht wird einfach zum Licht aus der vorigen Funktion addiert. Glanzlichter in der realen Welt sind üblicherweise weiß, da die Sonne und auch die meisten Lampen weißes Licht aussenden. In dem Fall sind Lichtfarbe.rot ... Lichtfarbe.blau alle 1.0.

Die obigen Berechnungen müssen für jede Lichtquelle getrennt ausgeführt werden. Die Farben werden dann addiert und ergeben in der Summe den direkten Lichtanteil, der an dem betreffenden Punkt auf der Objektoberfläche zu sehen ist. Dazu wird dann der Umgebungslichtanteil gezählt. Anschließend schickt man Strahlen in Reflektions- und Brechungsrichtung. Sie liefern den indirekten Lichtanteil.

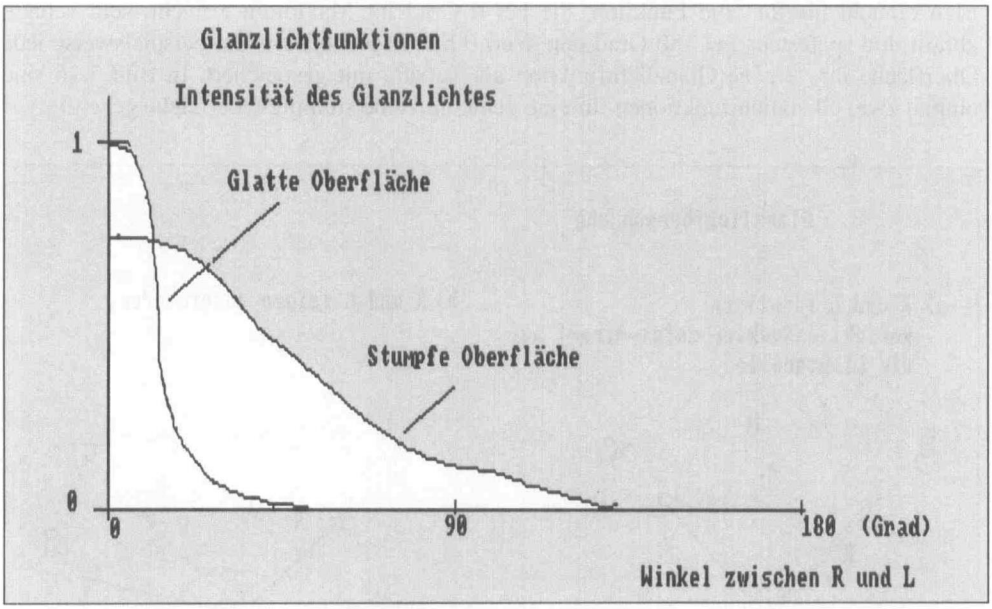


Bild 7.26: Glanzlichtfunktionen

Jetzt kommen die Gewichtungsfaktoren ins Spiel. Ein Material hat für direktes Licht, spiegelnden und brechenden Lichtanteil jeweils Faktoren, mit denen diese gewichtet werden. Die Anteile werden dann folgendermaßen zusammengezählt:

DFarbe = direktes Licht
SFarbe = Licht vom Spiegelstrahl
BFarbe = Licht von gebrochenem Strahl
UFarbe = Umgebungslicht
Gesamtfarbe = dir_faktor * DFarbe + spieg_faktor * SFarbe + brech_faktor * BFarbe + UFarbe

Ich habe hierbei nicht mehr die Rot-, Grün- und Blauanteile getrennt, die Rechnung muß für jeden Anteil extra durchgeführt werden.

Mit diesen Erläuterungen ist die Beleuchtungsfunktion – so hoffe ich – klar geworden. Vor allem bei Definition eigener Materialien dürfte dieses Kapitel nützlich sein. In »Construct« bestehen vielfältige Möglichkeiten, auf Materialeigenschaften Einfluß zu nehmen. Folgende Eigenschaften eines Materials können dort frei gewählt werden:

- Grundfarbe
- Anteile von direktem-, spiegelndem, brechendem und Umgebungslicht. Zusätzlich kann ein eigenleuchtender Lichtanteil des Materials gesetzt werden.
- Die Glanzfunktion kann frei editiert werden.

7.9 Antialias

»Alias« nennt man die Treppenstufen, die auf einem Rasterschirm auftreten, wenn man es mit Kanten zu tun hat, die entweder sehr flach oder sehr steil sind. Man sieht sie immer, wenn auf dem Bildschirm Linien gezeichnet werden, aber auch entlang von Objektkanten bei beliebigen Bildern. Dieser Effekt kommt von der relativ niedrigen Auflösung von Computer-Bildschirmen, verglichen mit dem menschlichen Auge oder mit einer Photographie. Ein Pixel auf dem Bildschirm ist eben kein Punkt, sondern ein Rechteck.

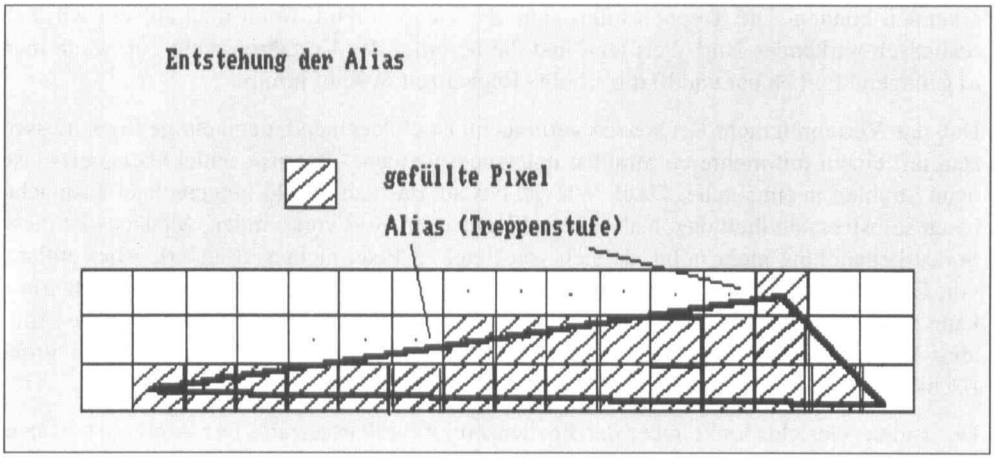


Bild 7.27: Die Entstehung der Alias

Beim Raytracing werden vom Auge aus durch alle Pixel Strahlen geschickt. Das Objekt, auf das ein Strahl als erstes trifft, wird dann im Pixel dargestellt. Der Strahl wird üblicherweise durch die Mitte des Pixels gelenkt, aber das ganze Pixel wird jeweils mit der ent-

sprechenden Farbe gefüllt, je nachdem, was der Strahl trifft. Liegt ein Objekt also so, daß der Strahl es knapp verfehlt, bekommt das Pixel nichts von der Farbe des Objektes ab, obwohl das Objekt eigentlich einen Teil des Pixels bedeckt. Erst wenn die Mitte des Pixels vom Objekt blockiert ist, wird das ganze Pixel gefüllt (Bild 7.27). Das auf diese Weise die Treppenstufen entstehen, leuchtet wohl ein.

Aber man kann etwas dagegen tun. Bei einer solchen Treppenstufe wird ein Farbsprung auftreten, das heißt, das Pixel wird von einem zum nächsten Pixel die Farbe oder die Helligkeit um einen relativ starken Betrag ändern. Wenn das nicht so wäre, würden dem Betrachter die Alias ja gar nicht auffallen.

Stellt das Raytraceprogramm also einen Farbsprung fest, der über einem bestimmten Grenzwert liegt, vermutet es eine Kante an dieser Stelle und schaut sich das betreffende Pixel genauer an. Das Pixel wird dann von mehreren Objekten teilweise bedeckt, und die Farbe muß korrekterweise aus allen Objekten zusammengemischt werden, die im Pixel vorkommen. Um das wenigstens annäherungsweise festzustellen, schickt der Raytracer nun ein ganzes Bündel von Strahlen in dieses eine Pixel. Jeder Strahl liefert eine Farbe zurück. Die werden dann zusammengemischt und liefern die endgültige Pixelfarbe.

Die Treppenstufe verschwindet dadurch an der Stelle. Die Kante erscheint glatt durchgezogen. Sie erscheint allerdings nur so, denn wenn man genau hinsieht, ist an der Stelle die Kontur unscharf bzw. verwischt. Das Verfahren kann also nicht die Mängel einer niedrigen Bildschirmauflösung beseitigen, sondern nur mildern. Wenn man im Bild Einzelheiten hat, die in der Größenordnung eines Pixels liegen, wird man die durch Antialias nicht genauer erkennen können. Die Treppenstufen sind aber sehr störend, wenn man auf ein wirklich realistisch wirkendes Bild Wert legt, und die beseitigt das Verfahren recht gut, wenn man a) genügend Farben hat und b) die erhöhte Rechenzeit in Kauf nimmt.

Daß das Verfahren mehr Rechenzeit verbraucht, ist einleuchtend, denn einige Pixel müssen statt mit einem mit mehreren Strahlen untersucht werden. »Beams« schickt beispielsweise neun Strahlen in ein solches Pixel. Wieviel das auf das ganze Bild umgerechnet ausmacht, hängt selbstverständlich davon ab, wieviel Kanten im Bild vorkommen. Meistens ist diese Sonderbehandlung nicht nötig, da sich von Pixel zu Pixel nicht viel ändert. Aber entlang von Objekträndern ist das Verfahren meist nicht zu vermeiden. Nach meiner Erfahrung kann man ungefähr mit einer Verdoppelung der Rechenzeit rechnen, wenn man das Antialias-Verfahren einschaltet. Es kann aber auch »schlimmer« werden, wenn Sie großflächige Texturen verwenden (siehe auch Kapitel 3.7 und 4).

Der andere Gesichtspunkt außer der Rechenzeit ist die Farbanzahl. Der Amiga ist zwar in seiner Klasse, die Farbanzahl betreffend, ungeschlagen, aber für realistische Bilder mit glatten Farbübergängen gibt es eigentlich noch viel zu wenige. Beim Amiga sind mehrere Bildschirmdarstellungsarten möglich, die sich in der Auflösung und in der Farbanzahl unterscheiden. Im HAM-Modus (Hold and Modify) sind zum Beispiel 4096 Farben möglich. Das reicht für viele Bilder schon aus, und hier lohnt sich das Antialias durchaus. In anderen Darstellungsarten sind aber zum Beispiel nur 32 oder 16 Farben möglich. Es gibt sicher Gründe, daß man ein Bild nur mit den wenigen Farben berechnen will, vielleicht

will man es in anderen Programmen weiterverwenden oder man braucht die hohe Auflösung, die nur 16 Farben ermöglicht – wie auch immer. Bei Verwendung weniger Farben würde ich das Antialias-Verfahren ausschalten, da sowieso zu wenig Farben da sind, um schon die normalen Objektfarben zu bilden. Für die Mischfarben an den Rändern reicht es da noch weniger.

7.10 Begriffsdefinitionen

In diesem Abschnitt werden zu verschiedenen Begriffen, die im Kapitel 7 vorkommen, jeweils einige Stichworte zur Erläuterung gegeben.

a. Objekt

Objekt ist ein recht allgemeiner Begriff, den man interpretieren kann. Im Zusammenhang mit der Bilderzeugung verstehen wir darunter eine mathematische Beschreibung eines bestimmten Körpers oder Gegenstandes im dreidimensionalen Raum. Eine Kugel zum Beispiel ist eindeutig durch die Koordinaten des Mittelpunktes und den Radius beschrieben, ein Dreieck durch die Koordinaten seiner drei Eckpunkte. Man könnte im Prinzip beliebig komplexe Objekte mathematisch beschreiben. So wäre beispielsweise ein Objekt »Haus« denkbar, das als Parameter Länge, Breite, Höhe, Winkel des Daches und noch weitere Größen enthält. Reflections verwendet aber als Objekte nur Dreiecke und Kugeln. Dies hat Vorteile beim Raytracing, da sich diese Objekte leicht bearbeiten lassen, aber auch beim Designen, da sich beliebig komplexe Körper aus diesen Grundobjekten zusammensetzen lassen.

b. Strahl

Ein Strahl im dreidimensionalen Raum ist durch einen Anfangspunkt und eine Richtung definiert. Für beides sind jeweils drei Komponenten nötig. Ein Strahl besteht demnach aus sechs Werten:

- *Anf_x, Anf_y, Anf_z* – die Koordinaten des Anfangspunktes.
- *Richt_x, Richt_y, Richt_z* – die Richtung des Strahls.

Anstatt der Richtung könnte man auch die Koordinaten eines Endpunktes nehmen.

c. Farbe

In der Computergrafik wird zur Farbdarstellung meist das RGB-Modell verwendet, weil auch die meisten Bildschirme die Farbe aus den Grundkomponenten Rot (R), Grün (G) und Blau (B) zusammensetzen. Jede Komponente bekommt eine Zahl zugewiesen, entsprechend ihrer Stärke. Bei der Farbe Weiß sind beispielsweise R, G und B alle auf volle Intensität gesetzt, bei Schwarz umgekehrt alle auf 0. Beim Amiga kann man zum Beispiel jeder Farbkomponente Werte zwischen 0 und 15 zuweisen. In professionellen Bilderzeugungs-

systemen gibt es je 256 Abstufungen, was dann insgesamt zu ca. 16 Millionen Farben führt (256*256*256).

d. Reflektionsgesetz der Strahlenoptik

Einfallswinkel = Ausfallswinkel. Der Einfallswinkel ist der Winkel zwischen dem einfallenden Strahl und dem Normalenvektor der Oberfläche (Bild 7.28).

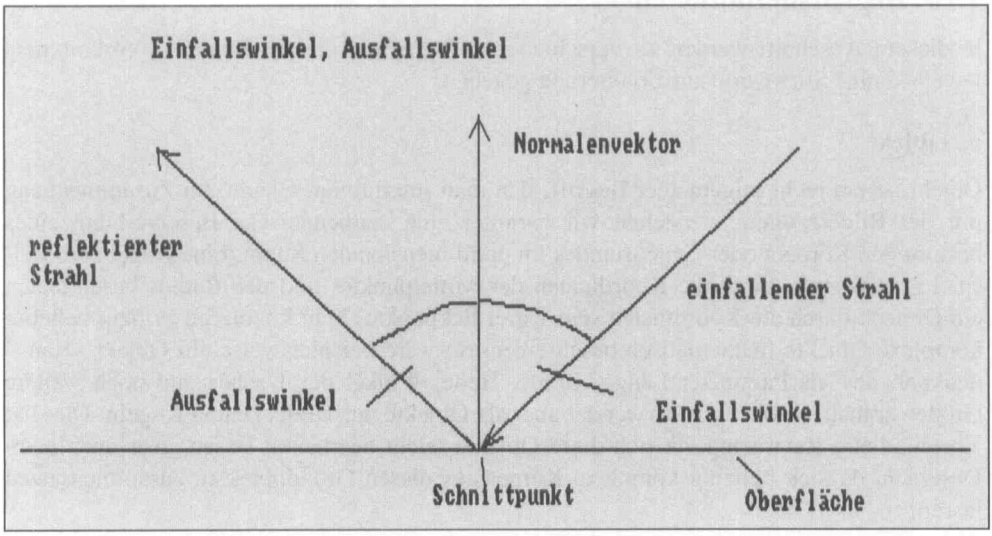


Bild 7.28: Das Reflektionsgesetz der Strahlenoptik

Der Normalenvektor steht immer senkrecht zur Objektoberfläche. Der Ausfallswinkel ist der Winkel zwischen dem Normalenvektor und dem reflektierten Strahl. Der Einfallswinkel wird bei Auftreffen des Strahls auf die Oberfläche berechnet.

e. Errechnung des Schnittpunktes eines Strahls mit einem Objekt
Beispiel: Objekt = Dreieck

Ein Dreieck ist geometrisch durch drei Eckpunkte im dreidimensionalen Raum beschrieben. Die drei Eckpunkte definieren eine Ebene. Die Ebene kann durch eine Ebenengleichung beschrieben werden, in der Form:

$a*x + b*y + c*z - d = 0$

a,b,c und d sind Werte, die genau die Lage der Ebene beschreiben. In diese Ebenengleichung kann man die Werte für einen Strahl einsetzen (Anfangspunkt, Richtung) und erhält den Schnittpunkt des Strahls mit der Ebene folgendermaßen:

(Strahl = (Anf_x,Anf_y,Anf_z, Richt_x, Richt_y, Richt_z)
*hilf = a*Richt_x + b*Richt_y + c*Richt_z*
*lambda = (d − a*Anf_x − b*Anf_y − c*Anf_z) / hilf*
*Schnitt_x = Anf_x + lamda*Richt_x*
*Schnitt_y = Anf_y + lamda*Richt_y*
*Schnitt_z = Anf_z + lamda*Richt_z*

In »Schnitt« sind dann die Koordinaten des Schnittpunktes. (Es muß natürlich getestet werden, ob »hilf« gleich 0 ist, in dem Fall ist der Strahl parallel zur Ebene und schneidet sie daher nicht.)

Nachdem der Schnittpunkt bestimmt ist, muß jetzt zusätzlich noch ermittelt werden, ob er auch innerhalb des Dreiecks liegt. Das Dreieck belegt ja nur einen kleinen Ausschnitt der Ebene. Dafür gibt es viele verschiedene Methoden. Eine einfache ist das sogenannte »Halbstrahl-Verfahren« (Bild 7.29).

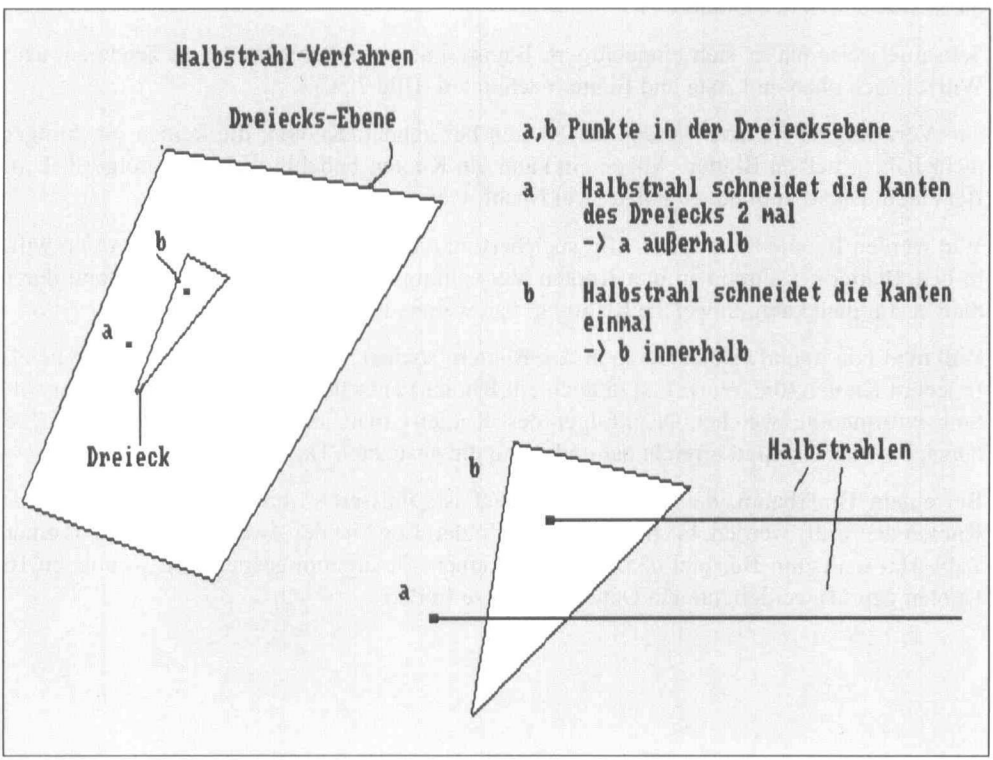


Bild 7.29: Das Halbstrahl-Verfahren

Bei dem Verfahren schickt man innerhalb der Dreiecks-Ebene vom Schnittpunkt einen Halbstrahl in eine beliebige Richtung. Dann prüft man nach, wieviele Kanten des Dreiecks er schneidet. Liegt der Schnittpunkt innerhalb des Dreiecks, dann schneidet der Halbstrahl genau eine Kante. Liegt er außerhalb, so sind es Null oder zwei Kanten (für allgemeine Vielecke liegt ein Punkt innerhalb, wenn die Anzahl geschnittener Kanten ungerade ist).

Übrigens, »lambda« ist das Maß, wieweit der Schnittpunkt vom Anfangspunkt des Strahls entfernt ist. Das Objekt, das den Strahl als erstes blockiert, ist das mit dem kleinsten lambda (lambda muß selbstverständlich größer als 0 sein, sonst liegt der Schnittpunkt hinter dem Anfangspunkt).

f. Baum

Bäume werden in der Informatik häufig benutzt, um Ordnung in große Datenmengen zu bringen. Der Name »Baum« kommt von der strukturellen Ähnlichkeit mit wirklichen Bäumen. Er hat eine Wurzel, die als Einstiegspunkt dient und verzweigt sich in viele Äste, die schließend in den Blättern enden.

Seltsamerweise hat es sich eingebürgert, Bäume immer verkehrt herum zu zeichnen, also Wurzel nach oben und Äste und Blätter nach unten (Bild 7.30).

Die Verzweigungspunkte werden als Knoten bezeichnet. Knoten, die keinen Nachfolger mehr haben, heißen Blätter. Allgemein kann ein Knoten beliebig viele Nachfolger haben. Bei einem Binärbaum hat er genau zwei Nachfolger.

Wie werden Bäume benutzt? Häufig speichert man die Daten, auf die man zugreifen will, in den Blättern, während in den Knoten Verwaltungsinformationen stehen, anhand derer man bestimmen kann, in welche Richtung man weitersuchen muß.

Will man nun irgendwelche Daten in den Blättern suchen, so beginnt man mit der Wurzel. In jedem Knoten (die Wurzel ist ja auch ein Knoten) entscheidet man anhand der Verwaltungsinformation, welchen Nachfolger des Knotens man aufsuchen muß. Das geht so lange, bis man ein Blatt erreicht hat und damit die gesuchten Daten.

Bei einem Binärbaum, der überall gleich tief ist, müssen so nur Log2 (Anzahl Daten) Knoten überprüft werden, bis man die Daten findet. Log2 ist der Zweier-Logarithmus einer Zahl. Hat man zum Beispiel 65536 Daten in einem Binärbaum gespeichert, so müssen 16 Knoten geprüft werden, um ein Datum daraus zu finden.

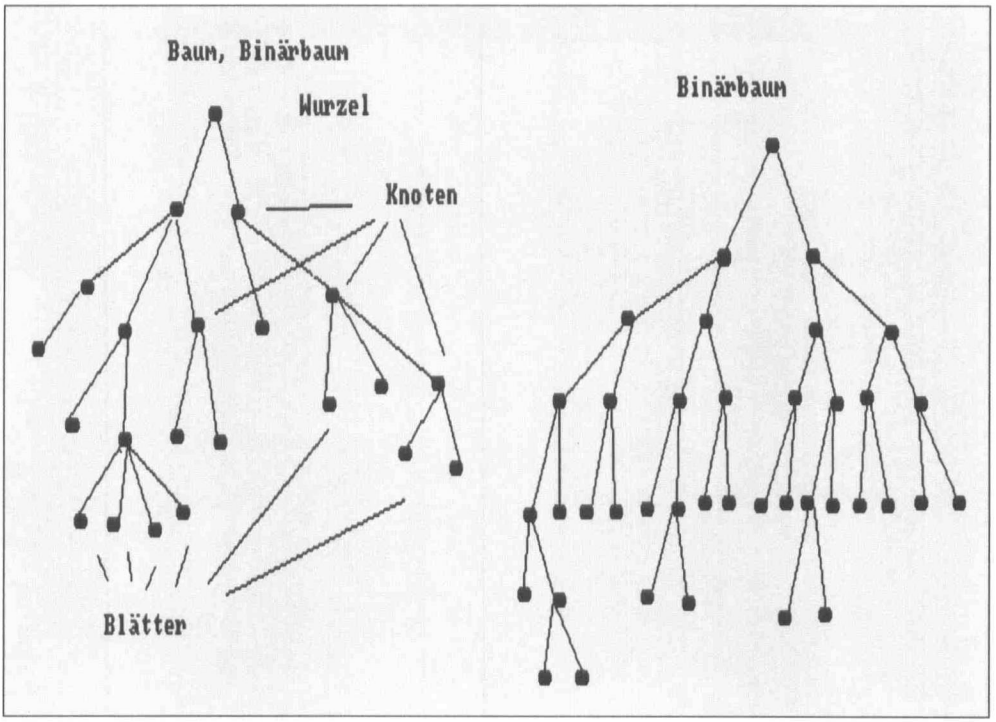


Bild 7.30: Baum, Binärbaum

Andere Algorithmen

In diesem Kapitel soll einmal ein kurzer Einblick in andere Verfahren zur Bilderzeugung gegeben werden. Raytracing ist ja nicht die einzige Methode, um realistische Bilder zu erzeugen. Im professionellen Animationsbereich wird das Raytrace-Verfahren wegen seiner großen Rechenzeit nur selten eingesetzt. Die Animationen, die in jüngster Zeit in Fernsehen und Werbung zu sehen sind, wurden größtenteils durch sogenannte Scan-Line Algorithmen erzeugt. Auch für Flugsimulatoren ist Raytracing vorläufig Utopie. Dort werden die Bilder tatsächlich in Echtzeit erzeugt. Dafür braucht man aber auch ganz spezielle Grafik-Rechner, bei denen der ganze Bilderzeugungsalgorithmus in Hardware implementiert wurde.

Bei vielen Einsatzgebieten ist Raytracing zudem gar nicht notwendig, jedenfalls nicht so, daß es die »Mehrkosten« durch die höhere Rechenzeit rechtfertigt. Bei einem Flugsimulator, bei dem in der Regel Landschaftsbilder zu sehen sind, wird Spiegelung bzw. Brechung eigentlich gar nicht gebraucht. Lediglich der Schattenwurf muß simuliert werden, da der Schatten dem Beobachter wichtige Informationen über die Lage von Gegenständen im Raum gibt. Auch im CAD-Bereich wird überwiegend noch mit anderen als dem Raytrace-Verfahren gearbeitet.

In ihrer Grundversion entsprechen die folgenden Verfahren dem Raytracing ohne Spiegelung, Brechung und Schatten. Wo diese drei Effekte nicht unbedingt nötig sind, kann man auf Raytracing verzichten, denn die anderen Verfahren sind dafür wesentlich schneller. Ein einleuchtender Grund für die Geschwindigkeit der anderen Verfahren ist die Ausnutzung der Kohärenz in einer Szene. Mit Kohärenz ist hier die Ähnlichkeit zwischen benachbarten Pixel gemeint. Meist ändert sich für ganze Bereiche von Pixeln nicht viel, beispielsweise für Pixel, die zum selben Objekt gehören. Das nutzt man aus, indem man diese Bereiche von Pixeln mit einem Schlag bearbeitet. Beim Raytracing fängt man hingegen bei jedem Pixel wieder bei Null an.

Ein anderer Grund ist eben die Tatsache, daß auf Schatten, Spiegelung und Brechung größtenteils verzichtet wird. Es existiert daher nur eine »Vorzugsrichtung« in der Szene,

nämlich vom Augenpunkt durch die Bildebene in die Szene. Das wird meist ausgenutzt, indem die Objekte in irgendeiner Weise in Bezug zu dieser Richtung sortiert werden, was dann die Bildberechnung stark beschleunigt. Beim Raytracing gibt es solch eine Vorzugsrichtung nicht, da Schatten-, Spiegel- und Brechstrahlen ja in beliebige Richtungen geschickt werden. Deswegen ist Raytracing auch dann noch langsamer als andere Verfahren, wenn man auf die Spezialeffekte verzichtet.

8.1 Der Maler-Algorithmus

Dies ist ein sehr einfaches Verfahren. Wie der Name schon sagt, ist das Verfahren der Arbeitsweise eines Malers nachempfunden. Der malt in sein Bild zuerst den Hintergrund, dann die Objekte, die hinten liegen, und schließlich diejenigen, die im Vordergrund sind.

Ganz grob kann man dafür folgenden Ansatz angeben:

- Sortiere alle Objekte nach aufsteigender Entfernung zum Beobachter.
- Fülle den Bildschirm mit der Hintergrundfarbe.
- Zeichne alle Objekte in der Reihenfolge von hinten nach vorne.

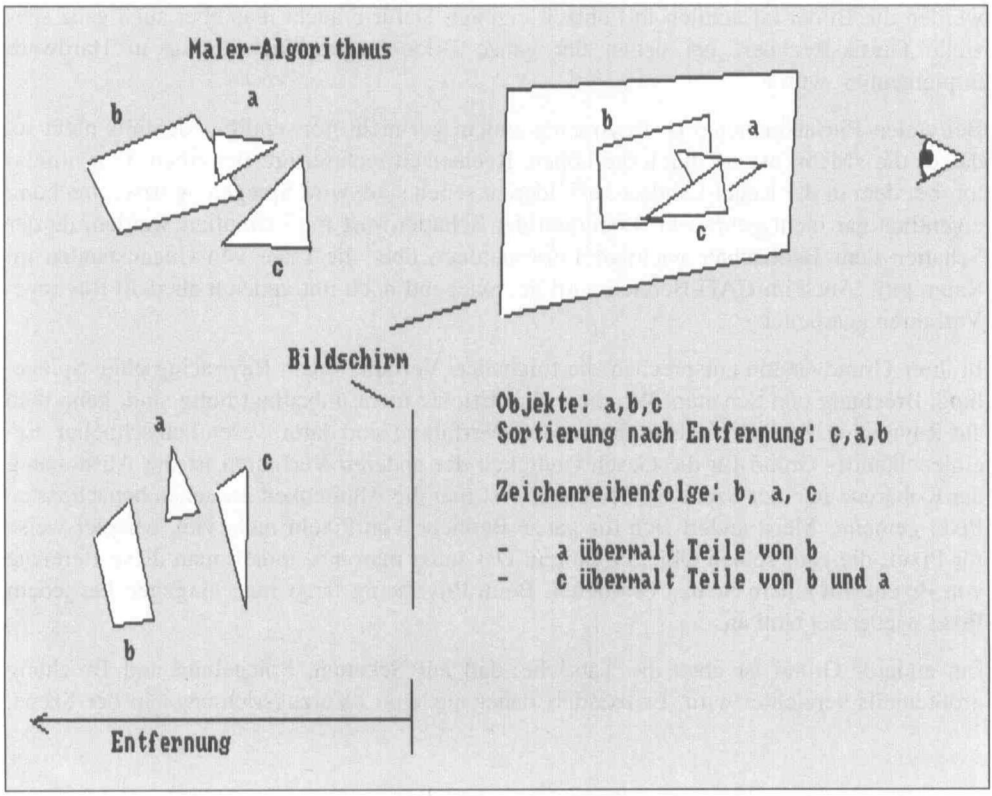


Bild 8.1: Der Maler-Algorithmus

8.2 Der Tiefenpuffer-Algorithmus

Er ähnelt im Prinzip dem Maler-Algorithmus in der Hinsicht, daß hintenliegende Objekte von vorderen übermalt werden. Nur ist hier keine Sortierung der Objekte notwendig, und die Priorität wird hier nicht objektweise, sondern pixelweise berechnet. Dieses Verfahren ist zwar ein sehr schnelles, aber es verbraucht auch sehr viel Speicher. Man braucht dazu folgende zwei zweidimensionale Felder (arrays), die genauso viele Einträge haben wie der Bildschirmpixel:

- *TIEFE*: Hier ist für jedes Pixel die aktuelle Entfernung zu dem Objekt gespeichert, von dem das Pixel im Augenblick bedeckt ist.
- *FARBE*: Hier ist die aktuelle Pixelfarbe gespeichert.

Der Algorithmus lautet folgendermaßen:

```
- Für alle Pixel des Bildschirms:
   {
   - Initialisiere TIEFE(Pixel) auf einen Wert größer als
     der größte mögliche Abstand zum Beobachter.

   - Initialisiere FARBE(Pixel) auf die Hintergrundfarbe
   }
    - Für alle Objekte der Szene:
   {
       - Für alle Pixel, die vom Objekt bedeckt werden:
      {
         - Z = Abstand des Objektpunktes, der das Pixel
               belegt, vom Beobachter.
         - Falls Z < TIEFE(Pixel) dann
         {
            TIEFE(Pixel) = Z
            F = Farbe(Objektpunkt)
            FARBE(Pixel) = F
         }
      }
   }
```

Durch den Vergleich mit TIEFE(Pixel) ist gewährleistet, daß immer nur die Farbe eines Objektes, das tatsächlich vor allen bisherigen Objekten liegt, für ein Pixel genommen wird. Man braucht sich auch nicht um Durchdringungen bzw. andere Problemfälle zu kümmern. Der einzige Nachteil des Verfahrens ist sein hoher Speicherverbrauch. Nimmt man beispielsweise eine Auflösung von 320*256 Pixeln, so braucht man allein für das Feld TIEFE mindestens 300 Kbyte. Bild 8.2 verdeutlicht das Tiefenpuffer-Verfahren.

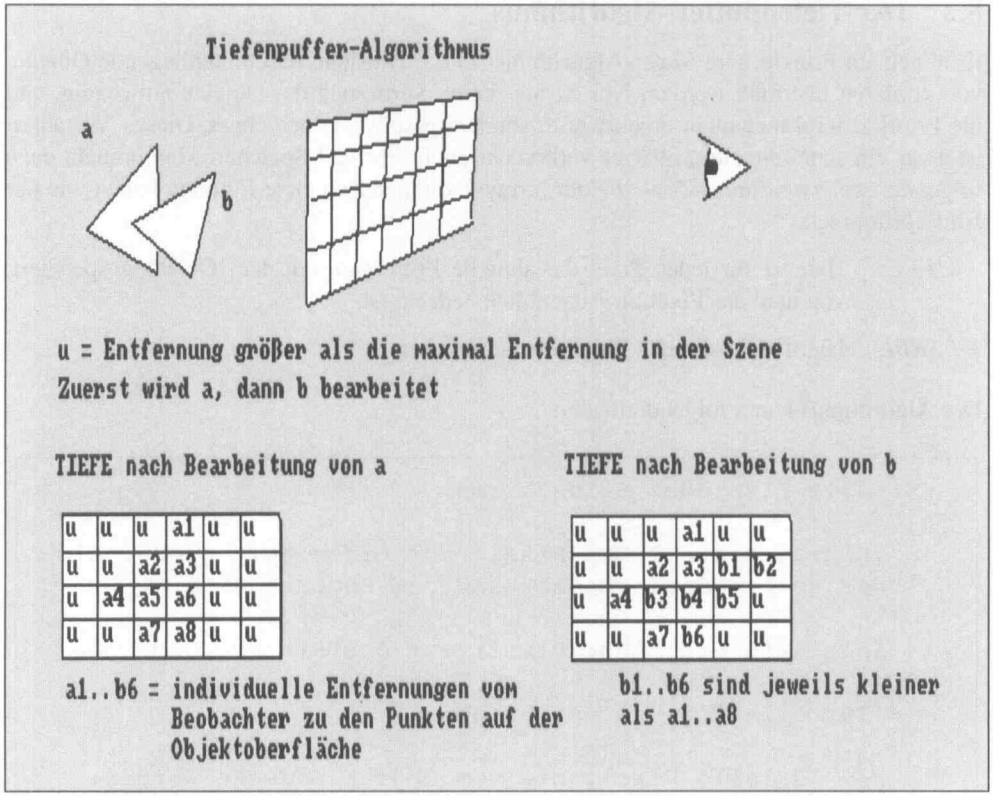


Bild 8.2: Tiefenpuffer-Algorithmus

8.3 Der Scan-Line-Kohärenz-Algorithmus

Dies ist eine Weiterentwicklung des Tiefenpuffer-Verfahrens, bei dem man den Nachteil, nämlich den hohen Speicherverbrauch, vermeidet. Das Prinzip dabei ist, den Bildschirm zeilenweise zu bearbeiten, damit müssen die Felder TIEFE und FARBE nur jeweils eine Bildschirmzeile aufnehmen.

Ganz einfach wäre es natürlich, den ganzen Tiefenpuffer-Algorithmus für jede Zeile getrennt auszuführen. Das würde aber einen großen Mehraufwand bedeuten, da für jede Zeile immer alle Objekte behandelt werden müßten, obwohl die meisten Objekte nur wenige Zeilen bedecken.

Es wird hier deswegen einiges an Vorverarbeitung vorgenommen, um in jeder Zeile nur Objekte bearbeiten zu müssen, die auch in der Zeile tatsächlich vorkommen. Man sortiert die Objekte daher mindestens in y-Richtung.

Dann wird eine sogenannte »Scan-Line«, eine Abtastlinie über die Objekte geführt. Die Scan-Line läuft im Bildschirm von oben nach unten. Dabei sind in einer Liste immer die Objekte gespeichert, die von der Scan-Line berührt werden.

Man muß jetzt in jeder Zeile nur eine kleine Objektmenge bearbeiten, nämlich die, die von der Scan-Line geschnitten werden.

Der Algorithmus lautet dann folgendermaßen:

```
- Sortiere alle Objekte nach y-Richtung und
  speichere sie in Y_Liste.
- Initialisiere SC_Liste auf leer.
- Für alle Bildschirmzeilen:
  {
    - Setze TIEFE für alle Pixel der Zeile auf den
      größten mögl. Wert.
    - Initialisiere FARBE auf Hintergrundfarbe.
    - Setze die Scan_Line auf die Bildschirmzeile.
    - Entferne alle Objekte aus SC_Liste, die
       oberhalb der Scan_Line liegen.
    - Füge alle Objekte aus Y_Liste in SC_Liste
      ein, die jetzt von der Scan_Line
      geschnitten werden.
    - Für alle Objekte aus SC_Liste:
      {
        - Für alle Pixel auf der Scanline, die vom
          Objekt bedeckt werden:
          {
            - Z = Abstand des Objektpunktes, der
                  das Pixel belegt, vom
                  Beobachter.
            - Falls Z < TIEFE(Pixel) dann
              {
                TIEFE(Pixel) = Z
                F = Farbe(Objektpunkt)
                FARBE(Pixel) = F
              }
          }
      }
    - Speichere FARBE ab.
  }
-fertig
```

Man sieht, daß dieser Algorithmus schon komplizierter als der Tiefenpuffer-Algorithmus ist. Ich habe auch nur eine grobe Skizze des Verfahrens gegeben, da gerade beim Scan-Line-Verfahren sehr viele Optimierungsmöglichkeiten bestehen, die dann allerdings das Ganze etwas unübersichtlich machen. So ist es zum Beispiel üblich, die Objekte in SC_Liste, also alle Objekte, die gerade von der Scan-Line geschnitten werden, auch nach x-Richtung sortiert zu halten und sie dann in x-Richtung nacheinander abzuarbeiten.

Das Einfügen der Objekte in SC_Liste geht auch recht schnell, da die Objekte ja bereits in y-Richtung sortiert vorliegen.

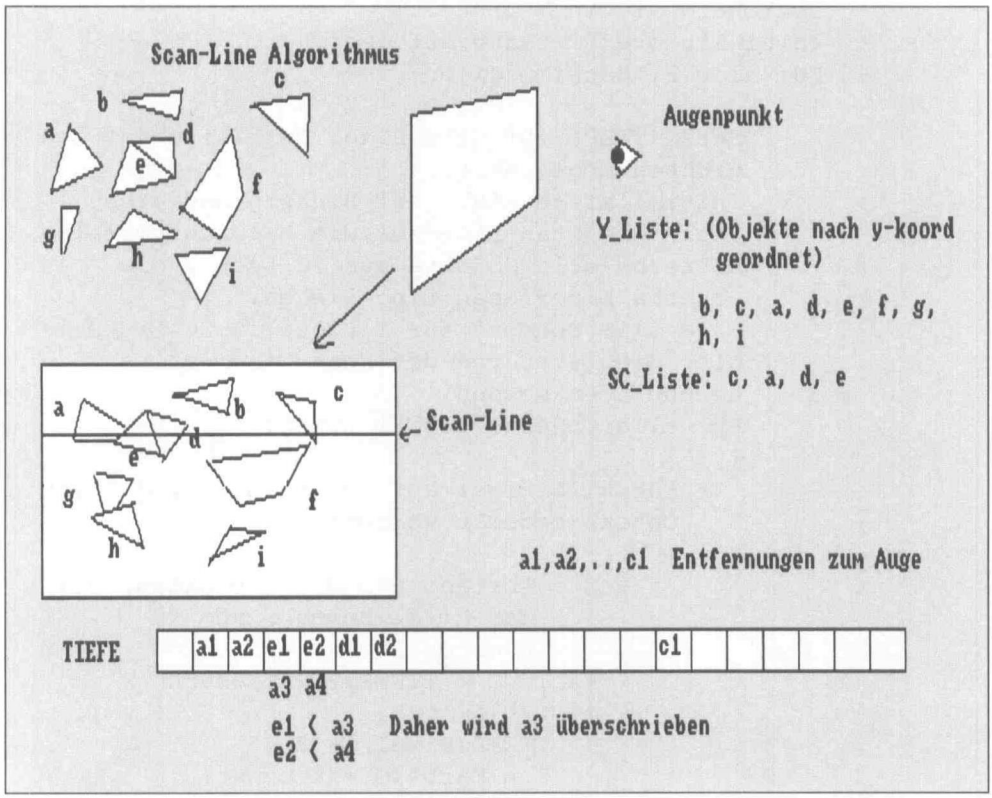


Bild 8.3: Scan-Line-Algorithmus

Gerade das Scan-Line-Verfahren wird bei professionellen Animationen am häufigsten eingesetzt, da es einen guten Kompromiß zwischen Geschwindigkeit und Speicherverbrauch bietet. Auch eignet es sich relativ gut für Parallelisierung, da hier lange Listen sortiert und in gleichartigen Schritten durchgearbeitet werden müssen. In vielen Animationsstudios werden daher auch schon Parallelrechner eingesetzt.

Einen weiterer Vorteil der obigen Verfahren liegt in der Antialias-Berechnung. Alias sind, wie Sie vielleicht aus dem letzten Kapitel noch wissen, die Treppenstufen, die wegen der geringen Bildschirmauflösung entlang von Kanten zu sehen sind.

Man behilft sich damit, daß man die Farbe in einem solchen Alias-Pixel entsprechend den Objekten, die es teilweise bedecken, mischt. Die Kante wird dadurch zwar etwas unscharf, aber die Stufen sind dadurch verschwunden. In den hier vorgestellten Verfahren »weiß« das Programm exakt (innerhalb der numerischen Genauigkeit), von welchen Objekten ein Pixel bedeckt ist. Daher kann im Alias-Fall die Farbe auch genau gemischt werden. Das Raytrace-Programm hingegen weiß gar nichts. Es probiert nur, indem es Strahlen in die Szene schickt. Das Auftreten von Alias-Pixeln merkt es nur, wenn Farbsprünge auftreten. Es schickt dann für dieses Pixel mehrere Strahlen los und mischt dann die zurückgelieferten Farben. Meist reicht das aus. Nur wenn Objekte auftreten, die auf dem Bildschirm dünner als ein Pixel sind, kann es gut vorkommen, daß diese Objekte übersprungen werden, d.h. von keinen Strahlen getroffen werden. Es entstehen dann häßliche Lücken im Bild. Das einzige Mittel dagegen ist nur die Erhöhung der Bildauflösung, was natürlich die Rechenzeit erhöht.

Ausblick

Reflections ist in erster Linie zur Erzeugung von Einzelbildern konzipiert. Einzelbilder sind zwar schön, aber bewegte Bilder wären noch schöner. Solche Animationen sind vor allem aus Fernsehen und Film bekannt. Es gibt kaum noch eine Sendung, die ohne Logo-Animation beginnt, wo also das Logo der Sendung möglichst spektakulär ins Bild »fliegt«. Auch beim Amiga sind Animationen inzwischen schon weit verbreitet. Allerdings sind Animationen von Raytrace-Bildern noch nicht so oft zu sehen. Wenn pro Bild mehrere Stunden gerechnet wird, bräuchte man pro Filmsekunde mindestens Wochen. Raytrace-Animationen sind bisher höchstens von recht einfachen Szenen zu sehen.

Ein anderes Problem ist die Definition des Bewegungsablaufes der Objekte. Mit »Construct« wäre das noch sehr umständlich. Man müßte jede Bewegung von Hand ausführen. Geplant ist aufgrund dieser Problematiken ein Erweiterungssystem zu Reflections mit mindestens folgenden Komponenten:

- Ein Programm zur bequemen Definition des Bewegungsablaufes von Objekten bzw. Körpern, Kameraflügen und der Änderung von Körpern in Abhängigkeit von der Zeit.
- Ein Programm zum Abspielen einer festgelegten Animation als Strichgrafik zur Kontrolle des Bewegungsablaufes.
- Ein weiteres Bilderzeugungsprogramm, das den Scan-Line-Kohärenz-Algorithmus verwendet (siehe Kap. 8.3). Damit sind zwar keine Spiegelungen, Brechungen und Schattenberechnungen möglich, aber dafür wird dann ein Bild in Minuten statt Stunden erzeugt. Denkbar ist auch eine Kombination mit »Beams«, bei dem »Beams« nur die Teile des Bildes berechnet, die spiegeln oder transparent sind. Der Rest des Bildes wird dagegen in Minuten erzeugt.
- Ein Programm zum Abspielen der erzeugten Bilder als Animation.

Literaturverzeichnis

J. F. Blinn Models of Light Reflection for Computer Synthesized Pictures, Computer-Graphics Vol.11(2), 1977, S.192

A. Fujimoto, K. Iwatawa Accelerated Ray Tracing. IEEE Computer Graphics & Applications, April 1986, S.16-25

C. Fuchs Raytracing großer Szenen mit Hilfe des Gitterbaums. Diplomarbeit am Institut für Informatik I, Universität Karlsruhe 1987

F. W. Jansen Data Structures For Ray Tracing, Graphics Interface 85, Springer Verlag 1986

M. Kadisch Raytracing komplexer Kugel- und Polygonszenen. Diplomarbeit am Institut für Informatik I, Universität Karlsruhe 1985

M. Linsenmann Raytracing komplexer Szenen. Diplomarbeit am Institut für Informatik I, Universität Karlsruhe 1986

H. Müller Computer-Graphics II, Computational Geometrie and Image Generation. Skriptum zur Vorlesung. Institut für Informatik I, Universität Karlsruhe 1984

Newmann, Sproull Grundzüge der interaktiven Computer-Graphik, McGraw-Hill 1986

A. Schmitt Graphische Datenverarbeitung I. Skriptum zur Vorlesung. Institut für Informatk I, Universität Karlsruhe 1984

T. Whitted An Improved Illumination Model For Shaded Display, CACM 23, 1980, S. 343

Stichwortverzeichnis

A

Abbruch 36, 88
Add 37
AddNeu 37
Antial 88
Antialias 88, 133
Auswahlfenster 20

B

Baum 138
Beams 83
Beleuchtungsformel 126
Benutzerschnittstelle 19
Bild 87, 97
Bilderzeugung, fotorealistische 7
Binärbaum 139
Blickrichtung 29
Brechung 110
Brechungsindex 50, 53
Brechungsvektor 126

C

CAD 7
Construct 13, 15
– starten 21

D

Darstellungsarten 29
Datei-Auswahlbox 21
Define 37
Def_Mat 43
Doppeln 39
Drehen 44, 46
Drehpunkt 46
Duplikat 39

E

Eckig 42
EditMat 50
Editob 52
Eigenleuchten 50, 53
Etik 59
Etikett-Textur 64

F

Farbanteil, brechender 53
–, diffuser 53
–, spiegelnder 53
Farbe 135
Farben 97
Farbmodus 29

Farb_0 32
Farb_1 32
Fenster 30
Flächentextur 57, 73
Frage-Fenster 21

G

Gen1 59
Genlock-Textur 66
Geomet. 16
Get_IFF 99
Get_sculpt 101
Gitterbaum 123
Gittermethode 120
Glanzlicht 130
Glanzlichtberechnung 131
Glanzlichtfunktion 132
Grid 84
Grundalgorithmus 104
Grundfarbe 49

H

Halbstrahl 137
HAM-Modus 89, 96
Hauptmenü 16
Hidd_0 31
Hidd_1 31
Hidd_2 32
Hinten 30
Hintergrund 72
HRB 35
HRD 35

I

IFF 63, 97
Interchange-File-Format 63

K

Kam.beweg 35
Kam/Li 86
Kamera 32
Kameraposition 32
Kantenmittelpunkt 47
Kegel 23
Knittern 44, 46, 48
Körper 16, 36
Körpername 43
Kugel 23, 36
Kugeltextur 57, 73
K_löschen 40

L

Laden 74
Licht 16
Licht, direktes 128
–, indirektes 128
Lichtbrechung 498
Lichtquellen 71
Lichtquellenvektor 126
Lichtweg 772
Licht_Farbe 71
Licht_Pos 71
Links 30
Listfenster 20
LoadKamLi 76
LoadMat 77
LoadSz 76

M

Maler-Algorithmus 142
Manager 88
Manager-Icon 13
Mat 60
mat/ 11
Material 16, 40
Material-Textur 67
Materialeigenschaft 16, 49
Matweg 52
Menü Geomet 18
–, Körper 17
–, Licht 18
–, Material 18
–, Plotten 17
–, Projekt 17
–, Tools 18
Modus 38, 88

N

NeuLicht 71
NeuMat 50
NeuOb 52
Normalvektor 126

O

Oben 30
Oberfläche 49
Oberflächenparameter 52
Objekt 15, 16, 135
Objekt-Konverter 101
Obweg 54
Octtree 125
OK 36
O_löschen 40

P

Parallelprojektion 30
Persp. 31
PKT 61
Plotfenster 19
Plotkörp 30
PlotModus 29
Plotten 16, 31
Plotten-Menü 14
Progr.Ende 77
Programm, interaktives 19
Projekt 16
Projekt-Menü 13
Projektion, perspektivische 31
Proportionalregler 20
Punkt-Textur 61, 68
Put_sculpt 102

Q

Quader 23
Quadtree 125

R

Raytrace-Parameter 86
Raytrace-Tiefe 87
Raytrace-Verfahren 103
Raytracen 104
Raytracing 7
Raytracing-Programmsystem 9
Ray_Tie 87
Rechenzeiten 92
Rechteck 20
Rechts 30
Reflektion, diffuse 49
–, spiegelnde 49
Reflektionsgesetz 136
Reflektionsvektor 126
RLB 35
RLD 35
Roll 36
Rotat.Körper 25
Rotationsachse 26
Rotationskörper 26
Rotationskörpertextur 57
Rund 42

S

SaveKamLi 76
SaveMat 77
SaveSz 76
Scan-Line-Kohärenz-Algorithmus 144
Schatt 87
Schatten 106
Schlagschattenberechnung 87
Schneiden 41
Schnitt.Krp 27
Schnitte 29
Schnittkörper 27
Sehstrahl 126
Show 95
Show-Parameter 97
Size 87
Skalierung 58
SP 87
Speichern 74
Speicherverbrauch 92
Spiegelung 108
Start 88
Strahl 135
Strahlenoptik 136
Strahlverfolgung 7

Sub 37
Suchbaum 118
Suche_Objekt 113
Szene 15, 86
Szene-Gadget 14

T

Tel 35
text/ 11
Textur 49
Textur-Objekt 55
Texturarten 57
Textureigenschaften 59
Texturen 54
Tiefenpuffer-Algorithmus 143
Tools 16
Treppenstufen 133

U

Umdreh 43
Umgebung 74
Umgebungslicht 128
Umgrenzungskörper 118
Unten 31
util/ 11

V

Verändern 44
Vergrößern 45
Verzerren 45
Vieleck 23
ViewMode 98
Vorne 30
VZB 35

Z

Zeigen 36
ZL 87
Zufallszahl 47
Zylinder 22

2-D-Bild 9
3D-Polygon 23

Amiga REFLECTIONS

Realistische Bilder generieren – ein Traum für Grafikfreunde, seit es Computer gibt. Anfangs war dieser Traum höchsten mit riesigen Computeranlagen und nach tagelangem Rechnen zu verwirklichen; heute wird er sogar für Heimanwender Wirklichkeit – vorausgesetzt, er hat die richtige Software.

Der Amiga erschließt aufgrund seiner Grafikfähigkeit aber auch Profis völlig neue Möglichkeiten, realistische Bilder mit Spiegelungen und Schattierungen zu entwickeln. Mit Reflections gilt das sogar für sehr komplexe Szenen in verblüffend kurzer Zeit: Setzen Sie Ihr Bild zunächst bausteinartig aus einfachen Einzelobjekten im komfortablen »Construct«-Editor zusammen, ordnen Sie diesen dann Materialeigenschaften wie Glas o.ä. zu oder projezieren Sie ein IFF-Bild auf eine Objekt-Oberfläche (Texture-Mapping); nun müssen Sie nur noch eine oder mehrere Lichtquellen setzen – »Beams« berechnet darauf das realistisch wirkende Bild mit über 16 Millionen Farben. »Show« wandelt es in eine Amiga-gerechte Darstellungsform um. Was dabei herauskommt, sehen Sie, wenn Sie sich einmal den Buchtitel oder den Farbteil im Buch anschauen. Erzeugen Sie also Bilder für Ihre private Diashow, Vorspänne für Ihre Videofilme oder Spitzengrafik für Ihre eigenen Programme!

Leistungsbeschreibung: Reflections erzeugt IFF-Grafiken und unterstützt den HAM-Modus, rechnet mit hoher Geschwindigkeit und ist multitaskingfähig. Das umfassende Begleitbuch macht die Bedienung nicht nur kinderleicht, es erklärt auch, wie Raytracing funktioniert und liefert Tips und Tricks für den Anwender.

BOOK-WARE

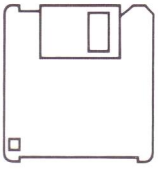
Profi-Software zum Buchpreis

Hardwareanforderungen:

Amiga 500, 1000 oder 2000 mit mindestens 512 Kbyte Speicherkapazität.

ISBN 3-89090-727-X

Markt&Technik



Unverbindliche Preisempfehlung

DM 98,–
sFr 94,10
öS 834,–